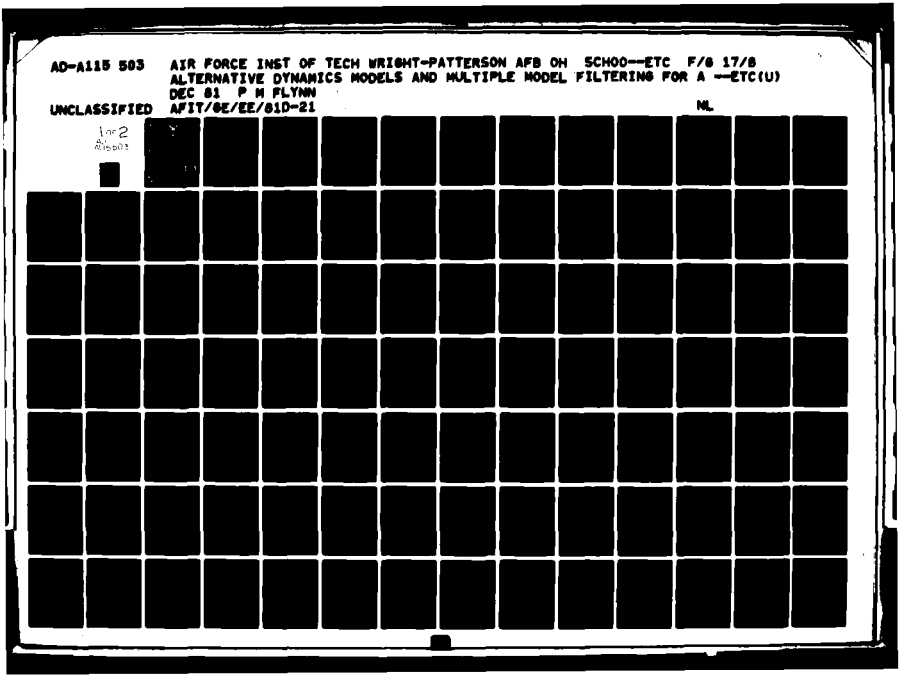
AD-A115 503 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 17/8
ALTERNATIVE DYNAMICS MODELS AND MULTIPLE MODEL FILTERING FOR A --ETC(U)
DEC 81 P M FLYNN

UNCLASSIFIED AFIT/GE/EE/81D-21 NL



DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/EE/81D-21

0

ALTERNATIVE DYNAMICS MODELS AND MULTIPLE MODEL FILTERING FOR A SHORT RANGE TRACKER

THESIS

AFIT/GE/EE/81D-21 PATRICK M. FLYNN

2Lt USAF

Approved for public release; distribution unlimited

AFIT/GE/EE/81D-21

ALTERNATIVE DYNAMICS MODELS AND MULTIPLE MODEL FILTERING FOR A SHORT RANGE TRACKER

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

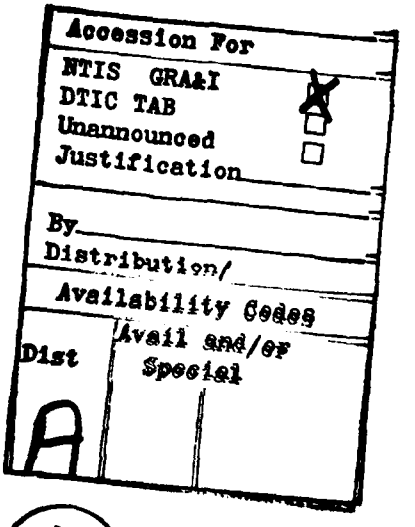
Patrick M. Flynn

2Lt USAF

Graduate Electrical Engineering

December 1981

Approved for public release; distribution unlimited



Acknowledgements

I wish to thank my wife for the support and encouragement she provided especially towards the end of this project when she stepped in and finished up all the typing even though she had her own studies to contend with and a new baby girl to take care of. I also want to thank my advisor, Professor Peter S. Maybeck, for all of his guidance and especially his patience. Without his help and his skill as a teacher, this project would not have been completed. Finally, I would like to thank the Air Force for giving me the opportunity to attain a Master's Degree so early in my career.

Contents

	Page
Acknowledgments	ii
List of Figures	v
List of Tables	vi
List of Important Symbols and Abbreviations	vii
Abstract	x
I. Introduction	1
Background	1
Problem	2
Previous Investigation	2
Assumptions	5
II. System Dynamic Models	8
General	8
Constant Turn Rate Dynamics Model	13
III. Multiple Model Filtering Algorithm	20
Theory	20
Multiple Model Algorithm Implementation	23
IV. Method of Evaluation	26
Monte Carlo Simulation	26
Trajectory Generation and Description	27
Figures of Merit	29
V. Results	34
General Performance of the Three Filters	34
Comparison of the Three Filters	36
VI. Conclusions and Recommendations	41
Conclusions	41
Recommendations	42
Computer Support	42

Contents (Cont'd)

		Page
Bibliography		44
Appendix A:	"F" Matrix Derivation	45
Appendix B:	Main Program and Library Listings	47
Appendix C:	Performance Plots for the Brownian Motion Filter.	86
Appendix D:	Performance Plots for the Constant Turn Rate Filter	119
Vita		160

List of Figures

Figure		Page
1	Elements of a High Energy Laser	3
2	Target Image in the FLIR field of view	4
3	Coordinate Frames	14
4	Multiple Model Filtering Algorithm	21
5	Typical Simulation Trajectory	28
6	Sample Output from Plotter Routine	31
7	Sample Output from Plotter Routine	32
8	Sample Output from Plotter Routine	33
C-1 to C-8	20 g Q=600 Performance Plots (BM)	87
C-9 to C-16	10 g Q=600 Performance Plots (BM)	95
C-17 to C-24	2 g Q=600 Performance Plots (BM)	103
C-25 to C-26	2 g Q=150 Performance Plots (BM)	111
D-1 to D-12	20 g Q=600 Performance Plots (CTR).	120
D-13 to D-24	20 g Q=300 Performance Plots (CTR).	132
D-25 to D-32	10 g Q=300 Performance Plots (CTR).	144
D-33 to D-40	2 g Q=300 Performance Plots (CTR).	152

List of Tables

Table		Page
1	Residual Statistics	38
2	Comparison of the Mean Error Magnitudes	39
3	Comparison of the Standard Deviation of the Errors	36

List of Important Symbols and Abbreviations

Symbol	Description
$\underline{A}$	Measurement residual covariance matrix
A_p	Area of picture element
BM	Brownian motion EKF
CTR	Constant turn rate EKF
EKF	Extended Kalman filter
$E[\cdot]$	Expected value
$\underline{f}(\cdot)$	Nonlinear system plant matrix
$\underline{F}$	Linear system plant matrix
$\underline{G}$	System noise input matrix
$\underline{h}(\cdot)$	Nonlinear measurement relation
$\underline{H}$	Linearization of intensity measurements
Hz	Hertz
$\underline{I}$	Identity matrix
I_F	Filter maximum intensity
$\underline{K}$	Kalman filter gain matrix
$\underline{P}$	Covariance matrix
$\underline{Q}$	Strength matrix of disturbance process
q	Noise strength
$\underline{R}$	Covariance matrix of measurement noise
$\underline{r}$	Residual vector
S/N	Signal to noise ratio: S/N = I_{max}/Background noise
t	Time
$\underline{v}$	General velocity vector

Symbol	Description
$\underline{w}$	White dynamics driving noise
$\underline{\hat{x}}$	Filter state estimate vector
$\underline{x}$	State vector
x_{peak}	Horizontal coordinate of Gaussian intensity function maximum
y_{peak}	Vertical coordinate of Gaussian intensity function maximum
$\underline{z}$	General measurement vector
$\underline{\Gamma}$	Vector of actual measurements
$\Delta\underline{x}$	Filter state vector update
Δx_v	Image plane velocity direction coordinate of intensity pattern
Δy_v	Coordinate perpendicular to image plane velocity of intensity pattern
θ	Orientation angle in image plane
σ_A	RMS value of atmospheric jitter (pixels)
σ_D	RMS value of target motion (pixels)
τ_A	Correlation time of atmospheric jitter
$\underline{\Phi}$	State transition matrix
	Subscripts
A	Atmospheric jitter
D	Target dynamics
F	Filter
o	Initial value
peak	Maximum intensity position
pv	Direction perpendicular to image plane velocity
v	Image plane velocity frame

Symbol	Description
	Superscripts
I	Inertial frame
T	Target frame
•	Time derivative
^	Estimate
+	After update
-	Before update

ABSTRACT

The performance of three extended Kalman filter implementations that estimate target position, velocity, and acceleration states for a laser weapon system are compared using various target acceleration trajectories. Measurements available to the extended Kalman filters each update are taken directly from the outputs of a forward looking infrared (FLIR) sensor. Two dynamics models considered for incorporation into the filter are 1) a Brownian motion (BM) acceleration model and 2) a constant turn rate (CTR) target dynamics model. The CTR filter was compared against the BM filter to see if the more complex dynamics of the CTR filter gave it a significant improvement in tracking performance over the BM filter. These two simple extended Kalman filters were then compared to a multiple model adaptive filter consisting of a bank of three filters based on the Brownian motion acceleration model. All three filters are tested using three different flight trajectory simulations: a 2 g, a 10 g and a 20 g pull-up maneuver. All evaluations are accomplished using Monte Carlo simulation techniques.

The constant turn rate extended Kalman filter was found to outperform the other two filters. The main advantage this filter had was the minimization of mean bias error in estimating position. The standard deviation of error was also slightly lower in most instances.

I. INTRODUCTION

Background

Low and moderate energy lasers have made important contributions in numerous applications including medicine, science, cartography, communications, range finding and target designation. The Air Force Weapons Laboratory (AFWL) located at Kirtland Air Force Base, New Mexico, has demonstrated the effectiveness of high energy lasers for use as a weapon in both stationary target, on-ground tests, and air-to-air tests using the airborne laser lab. Lasers have many unique characteristics, chief among these being the speed of light at which energy is transmitted from source to destination, virtually eliminating the need to "lead" the target. An aircraft flying at twice the speed of sound will only travel one-eighth inch in the time it takes laser energy to travel one mile.

Key components of a laser weapon system include both the laser itself, which generates the high power light, and the beam control subsystem, which aims the laser beam at the target and focuses it on a vulnerable spot on the target. The optics control (pointing) and target position estimation have to be very accurate in order to maintain the laser beam on a specific part of the target long enough to incapacitate a vital component of the vehicle. "Painting" the entire target with laser energy is inefficient and would require extremely high amounts of energy to achieve destruction. Elements of a high energy

laser system can be seen in Figure 1 (Ref 1).

The research in this report is concerned only with the "fine track" portion of the laser weapon system.

Problem

AFWL is interested in tracking air-to-air missiles at close ranges to accuracies better than achievable with the tracking algorithms currently in use. The objectives of this research were: (1)Design a Constant Turn Rate (CTR) dynamics model and compare the filter based on this truer acceleration model against a previously designed Brownian motion (BM) dynamics model. The purpose was to determine whether or not a substantial improvement in performance would be gained by going to the more complex acceleration model; (2)Design and test a multiple model adaptive filter for use as a laser tracker. The multiple model adaptive filter was designed to select the best tuned filter from a bank of filters to "fine track" a target in any one of a range of trajectories from flying straight and level to pulling high "g" turns. This bank of specialized or fine-tuned filters was then compared against the all purpose BM and CTR filters to determine whether or not a substantial performance improvement would be gained.

Previous Investigation

In 1978, Captain Daniel Mercier completed an initial thesis which demonstrated the feasibility of using an Extended Kalman Filter (EKF) designed to track benign targets using outputs from a forward looking infrared (FLIR) sensor as measurements. It exploits knowledge unused by current correlation trackers--size, shape, motion characteristics of the target, and atmospheric jitter spectral description to yield

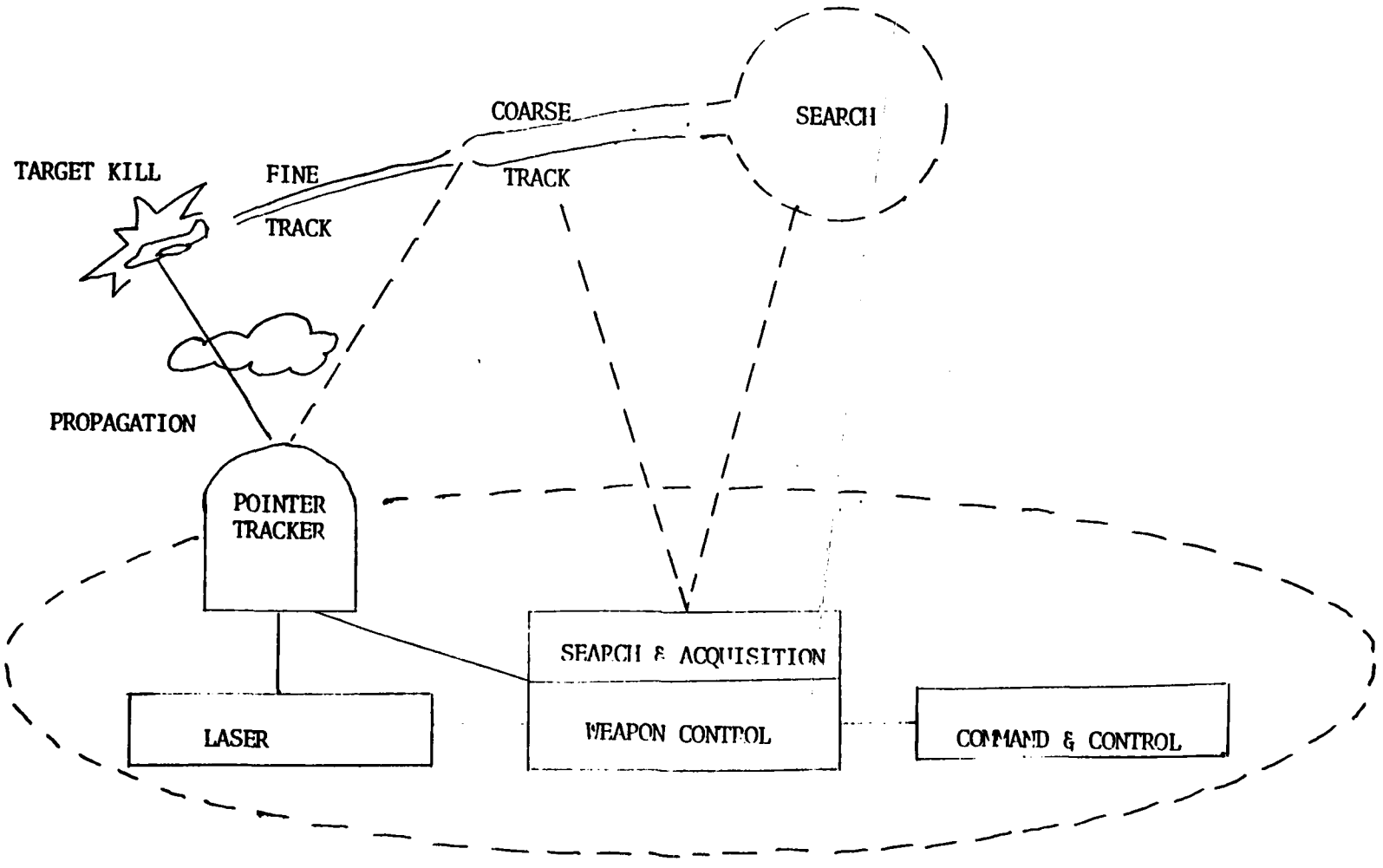


Figure 1. Elements of a High Energy Laser System

improved performance (Ref 2).

Captains Robert Jensen and Douglas Harnly continued the research in 1979. In order to track air-to-air missiles at close range, the algorithm they developed incorporated on-line adaption to target shape effects, changing target motion characteristics, and maximum signal intensity. The algorithm was shown to possess considerable performance potential for highly maneuverable targets despite background clutter by incorporating some ad hoc maneuver indicators.

Good tracking performance was achieved on the basis of estimating target velocity and acceleration as well as position. Elliptical constant intensity profile contours with major axis aligned with the estimated velocity vector were assumed (see Figure 2). Adaptively estimating the maximum intensity in the FLIR, the major and minor axis

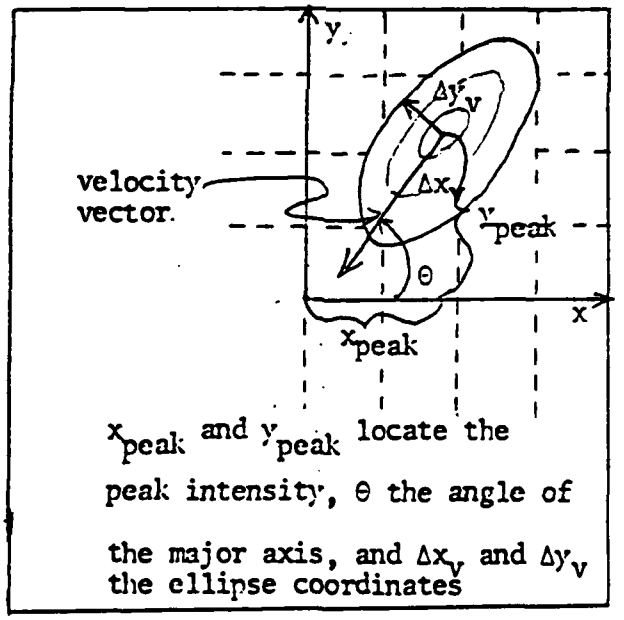


Figure 2. Target Image in FLIR Field of View

magnitudes of the ellipse, and the strength of the dynamic driving noise contributed to the good tracking performance that was achieved. Due to assumed zero-mean acceleration models, some rather tenuous ad hoc modifications were required to track the more dynamic targets (Ref 5).

The truth model and filter used by Jensen and Harnly (Ref 3) will serve as a beginning for this research.

Assumptions

FLIR. The FLIR outputs are instantaneous samples of an array of infrared detectors which are mechanically scanned by a many facetted mirror through a restricted field of view. Each minute infrared detector emits electrical current proportional to the average intensity of the infrared photons entering the face of the detector. A single digitized output represents a real time electronic spatial average of a horizontally scanned detector. The serial digitized data can either be stored or displayed on a cathode ray tube (CRT); each picture element of the CRT is called a pixel. The horizontal scanning of the detectors proceeds vertically through the FLIR field of view resulting in an array (frame) of pixels which is analogous in size (about 500 by 400 elements) and appearance to a normal TV picture. A new frame of pixels is generated every thirtieth of a second (30 Hz frame rate). For this study, an 8-by-8 array of pixels out of each large frame of pixels constitutes a single measurement array (often called a "tracking window") for processing by the proposed filter algorithms. Restricting the size of the measurement array is primarily dictated by computational and storage limitations and partially justified by fast measurement rates. (Ref 3:4-5)

Target. The target for some AFWL tests and this research is an air-to-air missile. A bivariate Gaussian intensity profile with elliptical equal-intensity contours with specified angular orientation is used. Real FLIR data supports the use of this model to approximate closely missile shape in the image plane (FLIR focal plane) (Ref 3:5).

The apparent target intensity function location in the FLIR field of view consists of several components. Boresight error, FLIR system vibrations and others are assumed to be negligible so that the intensity function location can be centered by:

$$\begin{bmatrix} x_{peak}(t) \\ y_{peak}(t) \end{bmatrix} = \begin{bmatrix} x_D(t) + x_A(t) \\ y_D(t) + y_A(t) \end{bmatrix}$$

where x_D, y_D = position offsets due to target dynamics

x_A, y_A = position offsets due to atmospheric jitter

Atmospheric jitter was modeled as a first order Gauss-Markov process, the output of a first order lag driven by white Gaussian noise, as follows:

$$\dot{x}_A(t) = -\frac{1}{\tau_A} x_A(t) + w(t)$$

where $w(t)$ is an independent white Gaussian noise process and τ_A is the correlation time of atmospheric jitter, 1/14.14 seconds. This model was used in both the x and y coordinate directions.

Background Noise. The image background noise which, along with FLIR sensor noise, contaminates the FLIR measurements, is modeled as a spatially and temporally correlated, Gaussian process. Various physical backgrounds result in FLIR images with differing spatial and temporal correlation characteristics. Real data analysis and AFWL experience

have been used to determine appropriate correlation coefficients for the spatial and temporal correlations.

Closed Loop. This is a closed loop tracking system. The laser pointing system is assumed to be perfect. That is, the system can point exactly where the tracker commands it within each sample period. This implies that settling time of the pointing system is less than the data sample period, the 1/30 sec time between discrete samples of the FLIR output (Ref 3:8).

II. SYSTEM DYNAMIC MODELS

General

Two different dynamic models were used in the bank of filters needed for the multiple model adaptive estimation algorithm. The first model, a Brownian motion acceleration model, was developed by Captains Harnly and Jensen (Ref 3). The second model, a constant velocity, constant turn rate model, was developed by Lieutenant Ruoff (Ref 6) and this author.

An extended Kalman Filter (EKF) algorithm was used for filter propagation and update since both filter types had nonlinear dynamics and/or measurement models. This algorithm provides a new reference state trajectory each time new state estimates are calculated.

Brownian Motion Dynamics Model. The Brownian Motion (BM) dynamics model development can be found in Harnley and Jensen's thesis (Ref 3) and will not be duplicated here. Only the results will be given.

The following BM dynamics model was used:

$$\underline{x} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \\ a_x \\ a_y \\ a_{t_x} \\ a_{t_y} \end{bmatrix} \qquad (1)$$

$$\dot{\underline{x}} = \begin{bmatrix} v_x \\ v_y \\ a_x \\ a_y \\ w_x \\ w_y \\ (-1/\tau_A)\, a_{t_x} + w_{A_x} \\ (-1/\tau_A)\, a_{t_y} + w_{A_y} \end{bmatrix} \qquad (2)$$

or

$$\dot{\underline{x}} = \underline{F}(t)\underline{x}(t) + \underline{G}(t)\underline{w}(t) \qquad (3)$$

where $\underline{F}(t)$ and $\underline{G}(t)$ can be deciphered from Eq. (2), and only the x and y coordinates are used since the data is taken off the FLIR image plane. where

x = azimuth position

y = elevation position

v_x = azimuth velocity = $\dot{x}$

v_y = elevation velocity = $\dot{y}$

a_x = azimuth acceleration = $\dot{v}_x$

a_y = elevation acceleration = $\dot{v}_y$

a_{t_x} = azimuth atmospheric disturbance (jitter)

a_{t_y} = elevation atmospheric disturbance

τ_A = correlation time of atmospheric jitter

w = zero mean white Gaussian noise

$$E[w_i(t)\, w_i(s)] = \sigma_i^2 \delta(t-s) \quad i = x, y, a_x, a_y$$

These eight state variables are related to the output measurements $\underline{z}$ by means of a nonlinear function.

$$\underline{z}(t_i) = \underline{h}\left[\underline{x}(t_i), t_i\right] + \underline{v}(t_i) \qquad (4)$$

where $v(\cdot)$ is a discrete-time white Gaussian noise sequence of zero mean and covariance kernel

$$E\left[\underline{v}(t_i)\, \underline{v}^T(t_j)\right] = \underline{R}(t_i)\delta_{ij} \qquad (5)$$

and $\underline{h}$ is a nonlinear function relating the states to the measurements. Specifically, for the jth of 64 pixels

$$h_j(\underline{x}(t_i),t_i) = \frac{1}{A_{pj}} \iint_{A_{pj}} I_F(t_i) \exp\{-\tfrac{1}{2} \Delta x_v \; \Delta y_v \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_{pv}^2 \end{bmatrix}^{-1} \Delta x_v \; \Delta y_v \; ^T\} \, dA_{pj} \qquad (6)$$

where

A_{pj} = area of pixel j, j=1, . . . 64

I_F = (filter estimate of) maximum intensity

$\Delta x = (x - x_{peak}) \cos\theta + (y - y_{peak}) \sin\theta$

$\Delta y = (y - y_{peak}) \cos\theta - (x - x_{peak}) \sin\theta$

θ = rotation angle between (x, y) and principal axis coordinates as shown in Figure 2.

σ_v = (filter estimate of) semi-major axis of ellipse in Figure 2. (v denotes along velocity vector)

σ_{pv} = (filter estimate of) semi-minor axis of ellipse in Figure 2. (pv denotes perpendicular to velocity vector)

Since the dynamics model is a linear time-invariant model, the time propagation relations are calculated as:

$$\hat{\underline{x}}(t_{i+1}^-) = \underline{\Phi}(t_{i+1},t_i)\hat{\underline{x}}(t_i^+) \qquad (7)$$

where $\underline{\Phi}(\cdot,\cdot)$ is the state transition matrix:

$$\underline{\Phi}(t_{i+1},t_i) = \exp(-\underline{F}\Delta t) \qquad (8)$$

for $\Delta t = t_{i+1} - t_i$ = constant sample period of 1/30 sec.

$$\Phi = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \Delta t^2/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \Delta t^2/2 & 0 & 0 \\ 0 & 0 & 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A1 \end{bmatrix} \tag{9}$$

$$A1 = \exp\left(-\Delta t/\tau_a\right)$$

The covariance propagation can be written as

$$\underline{P}(t_{i+1}^-) = \underline{\Phi}(t_{i+1},t_i)\ \underline{P}\ (t_i^+)\underline{\Phi}^T(t_{i+1},t_i) + \underline{Q}_d \tag{10}$$

where $\underline{Q}_d$ is the strength of the dynamic driving noise process matrix.

$$\underline{Q}_d = \int_{t_i}^{t_{i+1}} \underline{\Phi}(t_{i+1},\tau)\underline{G}(\tau)\underline{Q}(\tau)\underline{G}^T(\tau)\underline{\Phi}^T(t_{i+1},\tau)d\tau \tag{11}$$

$$\underline{Q}_d = \begin{bmatrix} A1 & 0 & A2 & 0 & A3 & 0 & 0 & 0 \\ 0 & A1 & 0 & A2 & 0 & A3 & 0 & 0 \\ A2 & 0 & A4 & 0 & A5 & 0 & 0 & 0 \\ 0 & A2 & 0 & A4 & 0 & A5 & 0 & 0 \\ A3 & 0 & A5 & 0 & A6 & 0 & 0 & 0 \\ 0 & A3 & 0 & A5 & 0 & A6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A7 \end{bmatrix} \tag{12}$$

where

$$A1 = \Delta t^5 \sigma_D^2/20 \qquad A5 = \Delta t^2 \sigma_D^2/2$$

$$A2 = \Delta t^4 \sigma_D^2/8 \qquad A6 = \Delta t \sigma_D^2$$

$$A3 = \Delta t^3 \sigma_D^2/6 \qquad A7 = \sigma_A^2 \left[1-\exp(-2\Delta t/\tau_A)\right]$$

$$A4 = \Delta t^3 \sigma_D^2/3$$

and σ_D is the RMS value of target motion and σ_A is the RMS value of the atmospheric jitter.

The measurement update relations for an EKF using the inverse covariance form are:

$$\underline{P}^{-1}(t_i^+) = \underline{P}^{-1}(t_i^-) + \underline{H}^T(t_i)\underline{R}^{-1}(t_i)\underline{H}(t_i) \tag{13}$$

$$\underline{P}(t_i^+) = \underline{P}^{-1}(t_i^+)^{-1} \tag{14}$$

$$\underline{K}(t_i) = \underline{P}(t_i^+)\underline{H}^T(t_i)\underline{R}^{-1}(t_i) \tag{15}$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i)\left[\underline{\Gamma}(t_i) - \underline{h}(\hat{\underline{x}}(t_i^-), t_i)\right] \tag{16}$$

Where $\underline{\Gamma}(t_i)$ is a sample of $\underline{z}$ at time t_i. $\underline{R}$ is the covariance of the measurement model and is a constant. The $\underline{H}(t_i)$ matrix is defined as:

$$\underline{H}[t_i;\underline{\hat{x}}(t_i^-)] \triangleq \frac{\partial \underline{h}[\underline{x},t]}{\partial \underline{x}}\Bigg|_{x=\underline{\hat{x}}(t_i^-)} \tag{17}$$

For details of how this derivation is computed, see Reference 3 and program listing (subroutine MEASF) found in Appendix B.

Constant Turn Rate Dynamics Model

The Constant Turn Rate (CTR) dynamic model differs from the Brownian motion dynamic model in the handling of the jerk level motion (time derivative of acceleration). The BM model expects zero mean acceleration with the jerk motion modelled as white noise. The CTR model replaces this jerk model with one that corresponds to a planar, constant velocity, constant turnrate maneuver, typical of the behavior of airborne targets.

The acceleration model was given by:

$$\dot{\underline{a}} = \omega^2 \underline{v} + \underline{w} \tag{18}$$

where

$\underline{v}$ = velocity in the FLIR image plane

$\underline{w}$ = zero mean white Gaussian driving noise

ω = magnitude of the target's turn rate; Eq. (28) shows how ω was calculated.

Eq. (18) was developed from the application of the Coriolis theorem written as:

$$\frac{I_{d(\underline{v})}}{dt} = \frac{T_{d(\underline{v})}}{dt} + {}^{I}\underline{w}^{T} \times {}^{I}\underline{v}^{T} \tag{19}$$

where "x" denotes the cross product and "I" and "T" represent the

inertial and target coordinate frames respectively. The FLIR image plane is treated as essentially inertial compared to the rotating target frame. The inertial frame origin is located at the tracker with "y" coordinate measuring altitude. The origin of the target frame is centered on the target (See Figure 3). On d( )/dt, an "I" or "T" superscript means as seen from that frame.

Note: ${}^{I}\underline{\omega}^{T}$ and ${}^{I}\underline{v}^{T}$ are the inertial target angular velocity and inertial target velocity respectively, and for notational simplicity are shortened to $\underline{\omega}$ and $\underline{v}$.

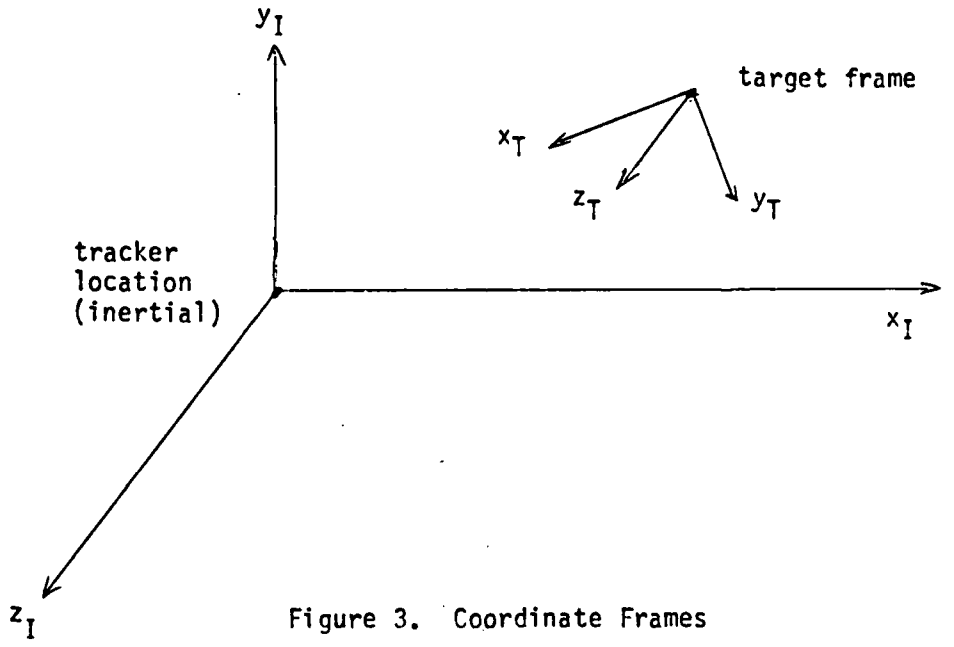


Figure 3. Coordinate Frames

Since the target is assumed to be of constant speed the first term on the right hand side is zero. Now the derivative of Eq. (19) with respect to time gives

$$ {}^{I}d^2\ (\underline{v})/dt^2 = {}^{I}d(\underline{\omega} \times \underline{v})/dt \qquad (20) $$

or expressed in the target body frame:

$$I_{d^2}(\underline{v})/dt^2 = T_d(\underline{\omega} \times \underline{v})/dt + (\underline{\omega} \times (\underline{\omega} \times \underline{v})) \tag{21}$$

Now since both target speed and angular velocity were assumed constant Eq. (19) becomes:

$$I_{d^2}(\underline{v})/dt = (\underline{\omega} \times (\underline{\omega} \times \underline{v}) \tag{22}$$

Using the relationship for a triple cross product, Eq. (22) can be written:

$$I_{d^2}(\underline{v})/dt^2 = (\underline{\omega} \cdot \underline{v})\underline{\omega} - (\underline{\omega} \cdot \underline{\omega})\underline{v} \tag{23}$$

The first term of Eq. (23) is zero since, for a planar, constant angular rate, constant speed turn, the target's inertial velocity and angular velocity vectors are perpendicular. Thus, Eq. (23) becomes:

$$I_{d^2}(\underline{v})/dt^2 = -\omega^2\underline{v} \tag{24}$$

which is the same as Eq. (18) with white noise to model the effects of modeling approximations and real world disturbances.

$$\omega = \left\| \frac{\underline{v} \times \underline{a}}{\|\underline{v}\|^2} \right\| \tag{25}$$

or equivalently:

$$\omega = \frac{\left| v_x a_y - v_y a_x \right|}{v_x^2 + v_y^2} \tag{26}$$

The filter states are the same eight states used in the BM model, but the dynamics model is now nonlinear.

$$\dot{\underline{x}}(t) = \underline{f}\{\underline{x}(t),t\} + \underline{G}\,\underline{w}(t) \tag{27}$$

$$\underline{x}(t) = \begin{bmatrix} v_x \\ v_y \\ a_x \\ a_y \\ -\omega^2 v_x + w_x \\ -\omega^2 v_y + w_y \\ -1/\tau_A + w_{A_x} \\ -1/\tau_A + w_{A_y} \end{bmatrix} \tag{28}$$

with ω given as in Eq. (26). The output measurements are taken as they were in the first filter.

The time propagation equations can be written as:

$$\underline{\hat{x}}(t_{i+1}^-) = \underline{\hat{x}}(t_i^+) + \int_{t_i}^{t_{i+1}} \underline{f}[\underline{\hat{x}}(t/t_i),t]\,dt \tag{29}$$

or since $\dot{\underline{\hat{x}}}(t/t_i) = \underline{f}[\underline{\hat{x}}(t/t_i),t]$ numerically integrating Eq. (29) using a first order Euler integration technique yields:

$$\begin{aligned} \underline{\hat{x}}(t_{i+1}^-) &= \underline{\hat{x}}(t_i^+) + \dot{\underline{\hat{x}}}(t_i)\Delta t \\ &= \underline{\hat{x}}(t_i^+) + \underline{f}[\underline{\hat{x}}(t_i^+),\, t_i]\,\Delta t \end{aligned} \tag{30}$$

This approximation is valid as long as Δt is small enough so

that second order terms and higher can be neglected. This first order approximation is also used in writing the $\underline{Q}_d$ matrix, as will be seen in Eq. (33).

The covariance propagation is the same as before only now $\underline{\Phi}(t_i+\Delta t, t_i)$ must be calculated every propagation:

$$\underline{\Phi}(t_i + \Delta t, t_i) = \exp(-\underline{F}(t_i)\Delta t) \tag{31}$$

which is a quasi-static approximation. It assumes that $\underline{F}$ is constant over the duration of the sample period. This approximation is good as long as the sample period is short as compared to the rate at which $\underline{F}$ changes. The $\underline{F}$ matrix is generated using the relation:

$$\underline{F}(t_i) = \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{\underline{x} = \underline{\hat{x}}(t_i^+)} \tag{32}$$

The upper left 6-by-6 partition of $\underline{\Phi}(t_i + \Delta t, t_i)$ is approximated with a first order Taylor series expansion. The correct closed form solution was easy to obtain for the two atmospheric disturbance states so truncating a Taylor series expansion to first order terms was not necessary. The closed form was used for those two states. The equation used for the elements associated with the first six states was:

$$\underline{\Phi}(t_i + \Delta t, t_i) = \underline{I} + \underline{F}(t_i)\Delta t \tag{33}$$

where

$\underline{I}$ = a 6x6 identity matrix

Δt = a sample period, 1/30 second

The derivation of $\underline{F}(t_i)$ can be found in Appendix A.

Since this program has to operate in real time with each sample period lasting under 1/30 sec (Δt), a number of simplifications in the form of first order approximations were made. As mentioned previously, the $\underline{Q}_d$ matrix was also simplified in this manner. This term represents the degree of uncertainty present in the filter propagation. In the case of a perfect dynamics model with no driving noise $\underline{Q}_d$ would be zero, reflecting absolute certainty in the estimate during the time between measurements. On the other hand, a poor dynamics model would require $\underline{Q}_d$ to have large values. The CTR model is believed to be a truer model of what the target dynamics are, so $\underline{Q}_d$ entries will be smaller; because of this and a short sample period, a first order $\underline{Q}_d$ matrix approximation is warranted. Thus, Eq. (14) is approximated by ignoring higher order terms to yield

$$\underline{Q}_d = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & A6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A7 \end{bmatrix}$$

where A6 and A7 are the same as before.

This approximation also avoids the need for recomputing $\underline{\Omega}_d$ each sample period, which would be necessary because $\underline{\Phi}$ changes each sample period (Eqs. (10), (32), and (33)).

III. Multiple Model Filtering Algorithm

Theory

The Multiple Model Filtering Algorithm is an adaptive filter composed of a bank of K separate Kalman filters, in this case, extended Kalman filters, each based on a particular value $\underline{a}_1$, $\underline{a}_2$, . . ., or $\underline{a}_K$ of the parameter vector (Ref 4: Ch 10, 103-110). The basic structure is shown in Figure 4.

This algorithm is based on magnitudes of time histories of residuals $\underline{r}_1(t_i)$, $\underline{r}_2(t_i)$, . . ., $\underline{r}_K(t_i)$ generated in the K filters when the measurement $\underline{z}_i$ becomes available. The residuals are processed by the hypothesis conditional probability computation shown in Eq. (34) and in turn the $p_k(t_i)$ values are used as weighting coefficients to generate $\hat{\underline{x}}(t_i^+)$ as seen in Eq. (35)

$$p_k(t_i) = \frac{f_{\underline{z}(t_i)|\underline{a},\underline{Z}(t_{i-1})}(\underline{z}_i|\underline{a}_k, \underline{Z}_{i-1})\, p_k(t_{i-1})}{\sum_{j=1}^{K} f_{\underline{z}(t_i)|\underline{a},\underline{Z}(t_{i-1})}(\underline{z}_i|\underline{a}_j, \underline{Z}_{i-1})\, p_j(t_{i-1})} \tag{34}$$

$$\hat{\underline{x}}(t_i^+) = \sum_{k=1}^{K} \hat{\underline{x}}_k(t_i^+)\, p_k(t_i) \tag{35}$$

where $\underline{x}_k(t_i^+)$ is the state estimate produced by a Kalman filter based on the assumption that the parameter vector equals $\underline{a}_k$. Therefore, the overall state estimate is the probabilistically weighted average of the state estimate generated by each of K separate Kalman filters.

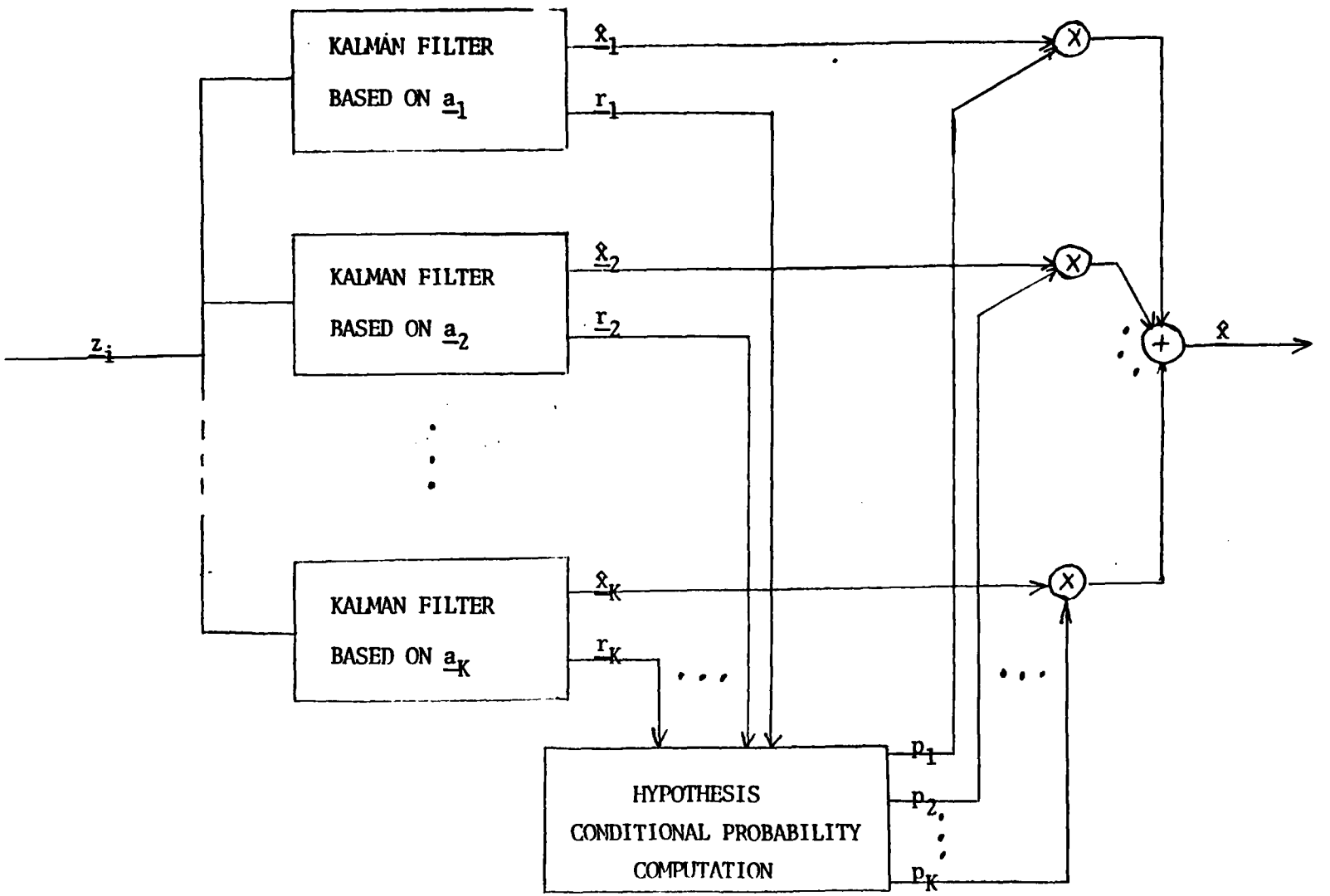


Figure 4. Multiple Model Filtering Algorithm

The conditional density function used in Eq. (34) can be evaluated as

$$f_{\underline{z}(t_i)|\underline{a},\underline{Z}(t_{i-1})}(\underline{z}_i|\underline{a}_k,\underline{Z}_{i-1}) = \frac{\exp\{\cdot\}}{(2\pi)^{m/2}|\underline{A}_k(t_i)|^{\frac{1}{2}}} \tag{36}$$

$$\{\cdot\} = -\tfrac{1}{2}\underline{r}_k^T(t_i)\underline{A}_k^{-1}(t_i)\underline{r}_k(t_i)$$

where $\underline{A}_k(t_i)$ is generated in the k-th Kalman filter as

$$\underline{A}_k(t_i) = \underline{H}_k(t_i)\underline{P}_k(t_i^-)\underline{H}_k^T(t_i) + \underline{R}_k(t_i) \tag{37}$$

and the residuals were calculated in each filter as

$$\underline{r}_k(t_i) = \underline{z}(t_i) - \underline{h}_k[\hat{\underline{x}}_k(t_i^-),t_i] \tag{38}$$

If it is desired to produce an estimate of the parameter vector itself, the conditional mean of $\underline{a}$ at time t_i is

$$\hat{\underline{a}}(t_i) = \sum_{k=1}^{K} \underline{a}_k p_k(t_i) \tag{39}$$

The residuals of the Kalman filter based upon the "correct" model are expected to be consistently smaller (relative to the filter's internally computed rms residual values) than the residuals of the other mismatched filters. If this is true, then equations (34) and (36) will cause the "correct" probability $p_k(t_i)$, i.e., the one whose index is associated with the correct filter model, to increase, while causing the others to decrease.

The performance of this algorithm is dependent upon significant difference between the residual characteristics in the "correct" and

the "mismatched model" filters. If the residuals are consistently of the same magnitude, then Eq. (34) and Eq. (36) result in the growth of the p_k associated with the filter with the smallest value of A_k which is independent of the "correctness" of the models and as such a result would be totally erroneous (Ref 4: Ch 10, 106).

Multiple Model Algorithm Implementation

Both the constant turn rate and Brownian motion acceleration models were considered for inclusion into the bank of filters required for the multiple model adaptive filtering algorithm. For the initial design and testing, it was decided to use only one type of filter in the filter bank. This decision was made because of the complexities involved in integrating two or more different types of filters into the program used and because of time constraints in completing this project. The CTR filter was found to be good at tracking targets over a wide dynamic range when properly tuned. This capability made it unsuitable for testing the multiple model algorithm as the residuals would be very similar for a bank of CTR filters. A Brownian motion filter tuned to a specific target trajectory had a smaller dynamic range than the CTR filter in which it outperformed other BM filters tuned to other trajectories. Thus, it lent itself better to this application than the CTR filter.

In this case, associated with each $\underline{a}_k$ was a different system model, differing in their assumed $\underline{Q}$ values. Three system models were used, three Brownian motion filters tuned to a 2g turn, a 10g turn, and a 20g turn.

$\underline{A}(t_i)$ is a 64x64 matrix and Eq. (36) calls for the inverse of

$\underline{A}(t_i)$ in the calculation of $p_k(t_i)$. This matrix inversion would use more computer time than available between updates, and so an approximation to $\underline{A}^{-1}(t_i)$ was made. One option considered to avoid a full matrix inversion was to use only the 64 elements of $\underline{A}(t_i)$ corresponding to the center 4 x 4 array of the FLIR (foveal area) since this is where the target image is expected to be reflected most of the time. The full inversion would then be taken of either of these 8x3 arrays and the elements then placed on the diagonal of a 64x64 array. The approximation chosen was to use only the diagonal of $\underline{A}(t_i)$, calculated in Eq. (37), in calculating the inverse. This was the easiest algorithm to implement and used the smallest amount of computer time. A full inversion would require more than 32 million multiplies for each filter every sample period, an 8x8 inversion requires approximately 6500 multiplies and about one thousand additions and subtractions. The 64x64 diagonal matrix chosen requires only 63 multiplies and 64 divisions. This approximation greatly decreases the amount of time required for each inversion.

The determinant of $\underline{A}_k(t_i)$ required in Eq. (36) would require over 8000 multiplies and so an approximation would be needed here too. Since all of the filters were of the same form, the $\underline{A}_k(t_i)$'s were similar while the major difference was in the residuals, so the scalar $(2\pi)^{m/2}\ \underline{A}_k(t_i)^{.5}$ terms were ignored in Eq. (34). The approximation that would have been used to evaluate $\underline{A}_k(t_i)$ also contributed to the decision to ignore the terms because of similarity.

Due to the size of the matrices involved, the product $\underline{r}_k^T(t_i)\ \underline{A}_k^{-1}(t_i)\underline{r}_k(t_i)$ often exceeded ± 480 which was too large to use as the

argument for the exponential function and exceeded the computer's upper numerical bounds. A scale factor of .01 was used to bring the product's magnitude down to acceptable levels. This also changes the magnitude of the ratio of one filter to another. This was assumed to be acceptable (Ref 7) until a better method of scaling the exponent could be found. The residuals from the "foveal region" and the optional 8x8 $\underline{\Lambda}(t_i)$ matrix mentioned above could have been used so that scaling might not have been necessary.

The final form of Eq. (30) implemented was:

$$f_{\underline{z}(t_i)\ \underline{a},\underline{Z}(t_{i-1})}(\underline{z}i\ \underline{a}k,\underline{Z}i\text{-}1) = \exp\{-.005\underline{r}_k^T(t_i)\underline{\Lambda}_k^{-1}(t_i)\underline{r}_k(t_i)\} \quad (40)$$

The p_k values from Eq. (34) were lower bounded to keep the probability factor from going to zero and to keep the reaction time, necessary to adjust to a new target trajectory, at acceptable levels. The minimum value of p_k was chosen to be .01. This choice was based on test results of the program itself.

IV Method of Evaluation

Monte Carlo Simulation

Modified versions of an existing digital simulation program (Ref 3) were used to test each of the three filter's performance. The propagation of the filter's state estimates and covariance portions of the program were modified to reflect either a Brownian motion or constant turn rate extended Kalman filter. Additionally, the program was modularized for the adaptive multiple model extended Kalman filter in order to take advantage of sections of the program that are common to all filters. This avoids having to duplicate large sections of code.

The outputs from this program, the truth model state vector, the filter's estimate of the state vector, and the filter's covariance matrix for each interval, were stored for post-processing. A statistical evaluation and plotting routine developed by Captain Mercier (Ref 2) was used to do the post-processing.

The number of Monte Carlo runs made through each simulated trajectory for each filter was 20. This number was selected to provide confidence in the accuracy of the solution. Captains Harnley and Jenson showed that 20 runs was sufficient in their thesis (Ref 3) and was verified for this work. The choice of 20 passes through each simulation also kept the computer execution time and storage requirements at acceptable levels.

The plots available from the plotting routine were 1) the mean error between the truth model (target trajectory data) value and the corresponding estimate from the filter, plus and minus the actual standard deviation as generated in the Monte Carlo analysis, 2) the actual standard

deviation versus the filter's estimate of the standard deviation, and 3) a variance convergence plot showing whether or not 20 Monte Carlo simulation runs were sufficient. This last plot showed the variance at four separate points in time in the simulation as each Monte Carlo run was made. As more passes were made through the simulation, the variance at each point in time leveled off indicating convergence.

The first two plots were used to evaluate filter tuning and performance. The first plot was used to check the mean bias and peak errors and the second plot was used to compare the magnitudes of the standard deviation.

Trajectory Generation and Description

The trajectories used in the evaluation of the three filters were all basically identical with only the force of the pull-up maneuver being varied. The trajectories, picked to represent typical target maneuvers, all started with a constant inertial velocity cross range path with a constant multiple 'g' pull-up initiated two seconds into the simulation. Each engagement lasted five seconds, providing adequate time for the performance evaluation of the filters. Figure 5 shows a typical trajectory.

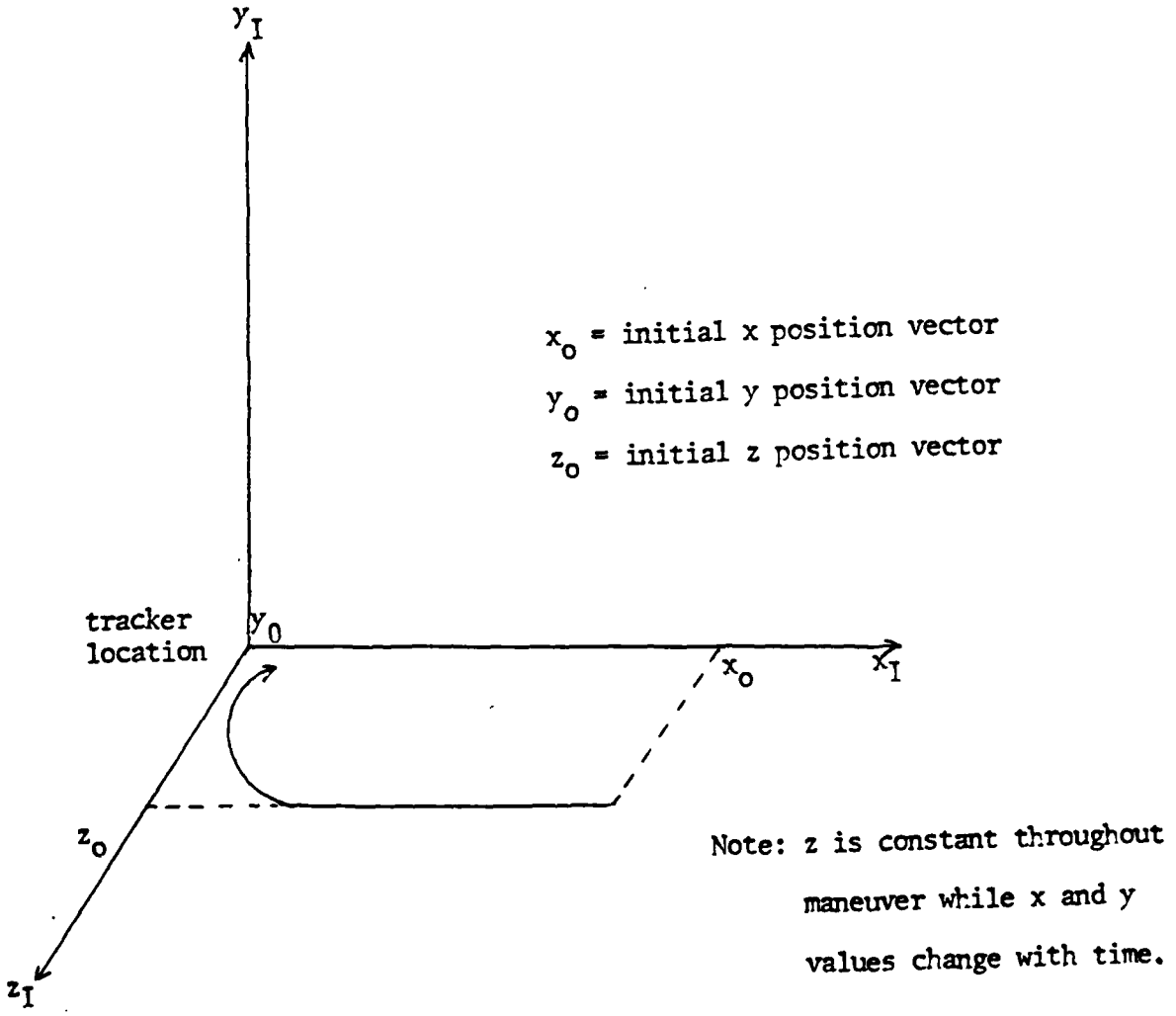


Figure 5. Simulation Trajectory

Before the pull-up, the equations are

$\dot{x} = -1000$ m/sec

$\dot{y} = 0$

$\dot{z} = 0$

$x = 6{,}000$ m

$y = 0$

$z = 40{,}000$ m

After pull-up initiation, the equations are

$\dot{x} = -1000 \cos \omega(t-2)$ m/sec

$\dot{y} = 1000 \sin \omega(t-2)$ m/sec

$\dot{z} = 0$

$x = -4000-(1000/\omega)\sin(\omega(t-2))$ m

$y = (1000/\omega)(1. - \cos \omega(t-2))$ m

$z = 40{,}000$ m

where ω equals 0.0196, .098, or 0.196 radians for turn magnitudes of 2 g's, 10 g's, and 20 g's respectively. These paths were intended to show the ability to track a highly maneuvering missile.

Figures of Merit

The plot routine printed out the mean error plus or minus the standard deviation of the error, and the filter estimate of the standard deviation versus the actual standard deviation for each sample period using the computer average of the 20 Monte Carlo runs. This was done for the first four error states, the x and y coordinate position and velocity errors. This data was then used to find the time averaged mean error and the time averaged variance over the last two seconds of the engage-

ment. The last two seconds of the engagement were used so that any initial transient effects caused by the change in dynamics of the maneuver would have diminished. This provided 60 sample points (1/30 second sample period), each point being an average of 20 Monte Carlo runs, to use in calculating the time averaged statistics. Sixty points was felt to be enough because of the Monte Carlo technique used in obtaining each point. Also, the data was used to find the peak error after the pull-up maneuver was initiated.

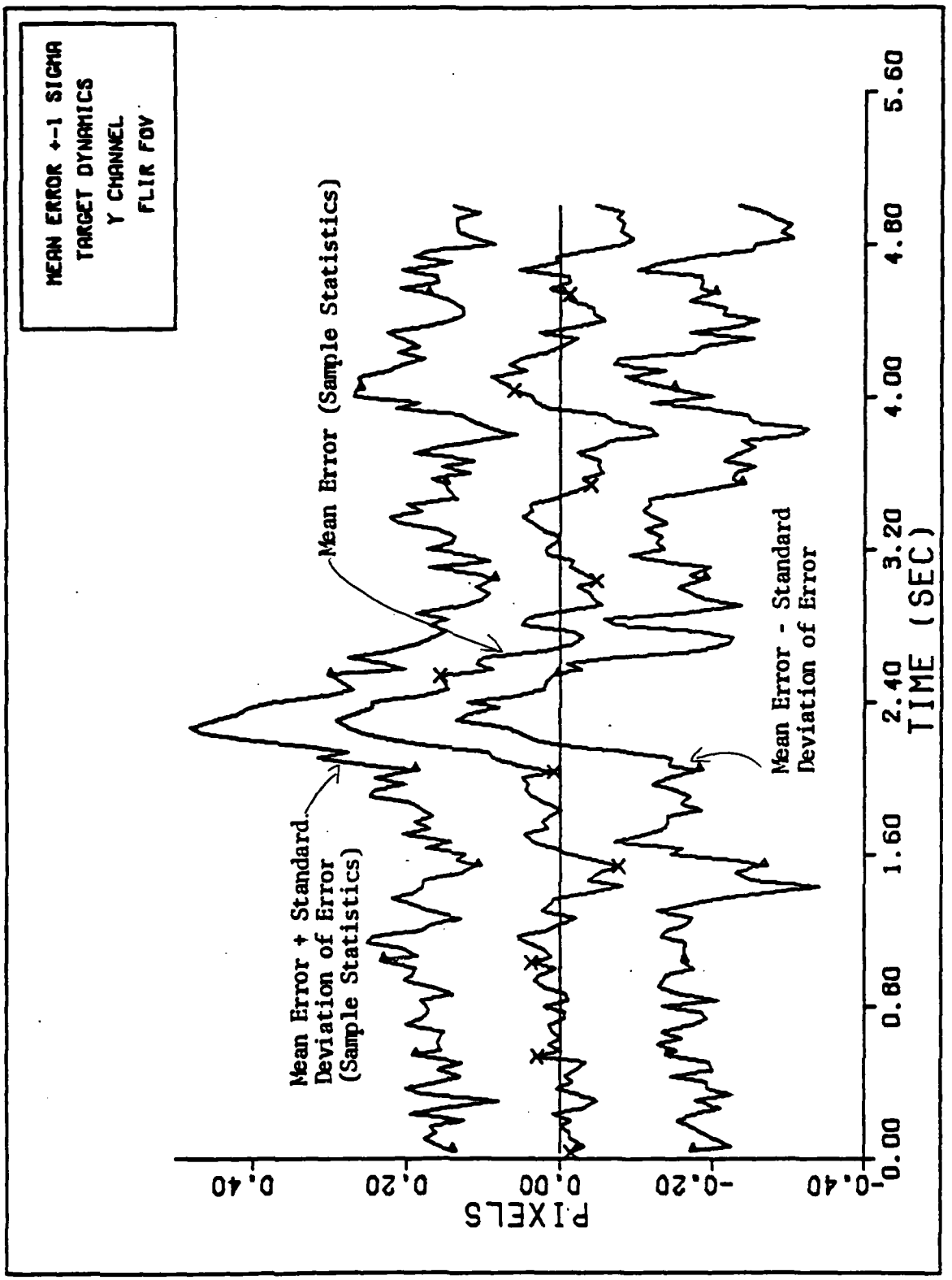
Again, the three figures of merit calculated for the comparison of the Brownian motion, the constant turn rate and the adaptive multiple model filters were:

1) the time averaged value of the x and y mean errors in position and velocity (all rounded to three decimal places);

2) the peak value of the x and y mean errors in position and velocity (all rounded to three decimal places); and

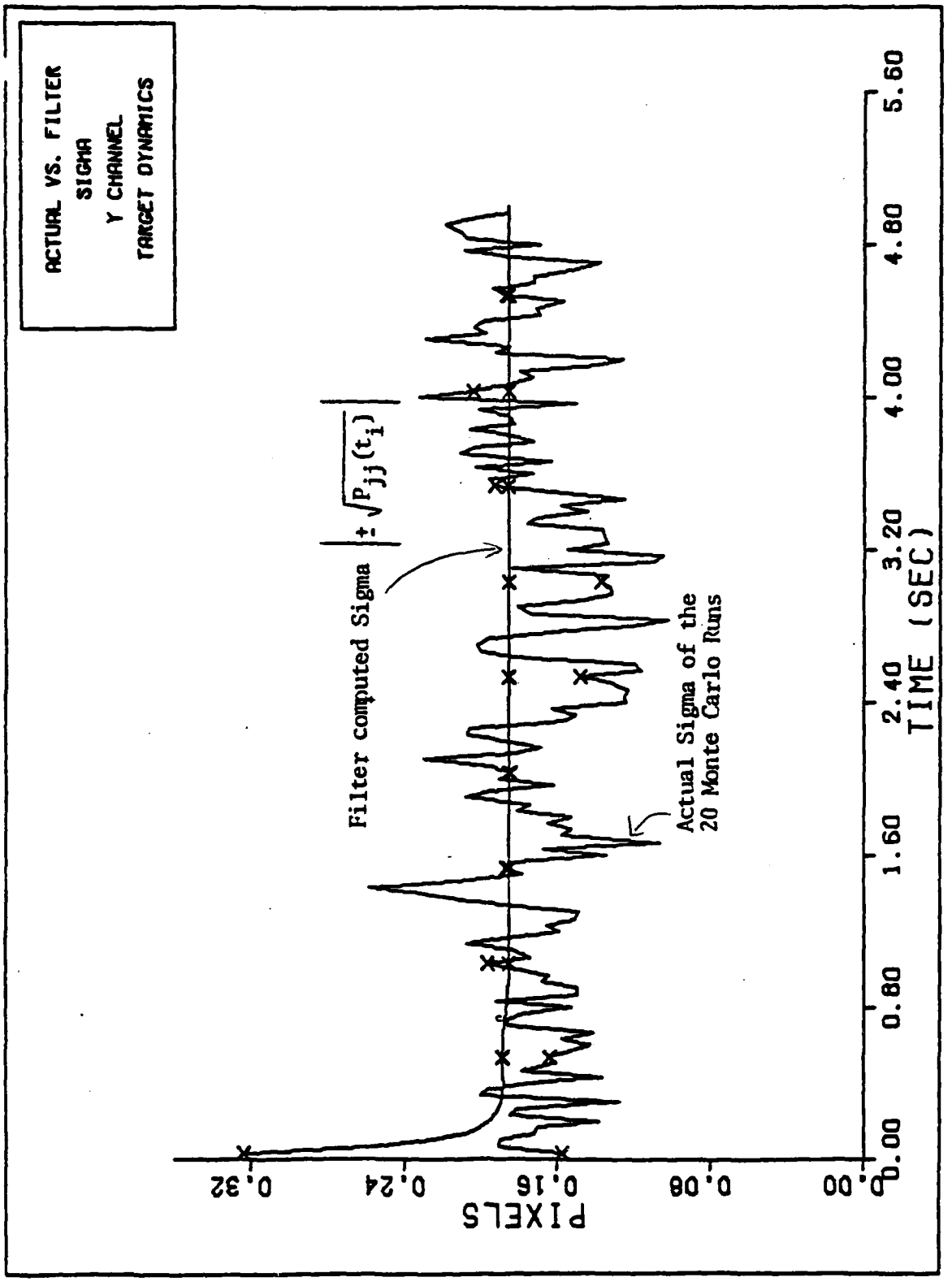
3) the time averaged value of the standard deviation of the errors committed by the filter for the x and y position and velocity (all values rounded to three decimal places).

Samples of the plots used in tuning the Brownian motion and constant turn rate filters can be seen in Figures 6, 7, and 8.



Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure 6. Sample Output from Plotter Routine



FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure 7. Sample Output from Plotter Routine

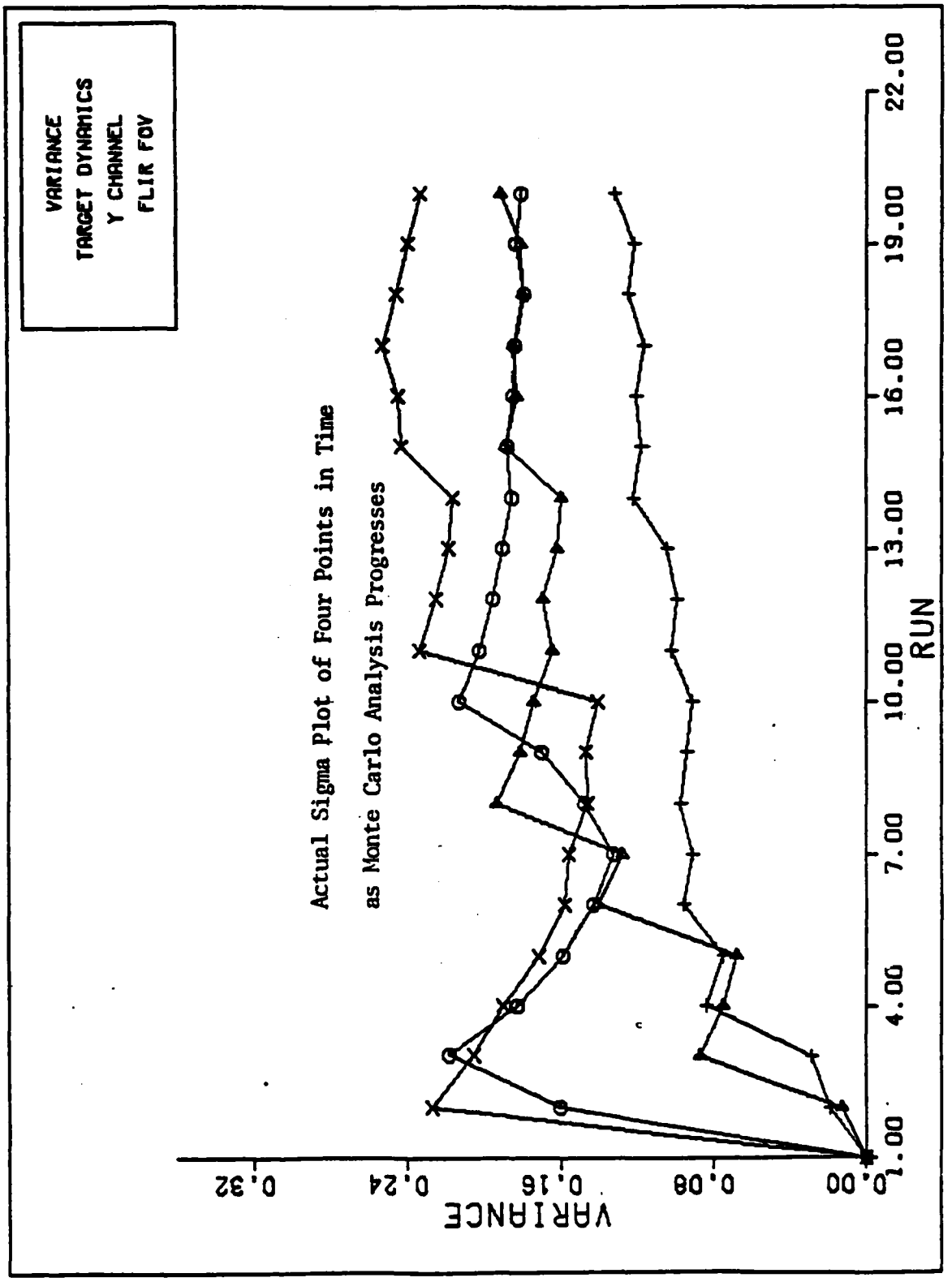


Figure 8. Sample Output from Plotter Routine

V Results

General Performance of the Three Filters

Before the direct comparison between filters is made, the general performance of each filter will be discussed to provide physical insight into the results. Figures C-1 to C-32 are the plots of the results from the Monte Carlo simulations of the Brownian motion (BM) extended Kalman filter for the three trajectories used. The BM filter was characterized by oscillatory, biased estimates in both of the FLIR x and y axes for position and velocity. The largest bias errors were in the x position and y velocity estimations. Also, the biases in the estimates got larger as the pull-up acceleration was increased. These large mean errors were a result of the inadequacy of the assumed target relative acceleration model. This model, a Brownian motion process, did not accurately model the target acceleration when the target dynamics were anything but straight and level flight as was the case for the first two seconds of all the simulations or if the acceleration was Brownian motion. Thus, the BM filter had an inherent lage in the estimates of the states when unmodeled maneuvers with persistent turning accelerations were encountered. The high measurement update rate is what kept the filter from diverging and losing track of the target while the unmodeled maneuver was occurring.

The mean errors of all the estimates of the states made large excursions outside the envelope of plus or minus the square root of the corresponding filter-computed conditional variance, once the pull-up simulation started. The large mean errors and the failure of the filter's conditional variance to reflect the growth of the mean errors

indicated that the filter was putting too much weight on the results from the dynamics model and not enough weight on the information contained in the measurements. Further tuning of the BM filter or the use of adaptive Kalman filter techniques (Ref 3) would help alleviate this problem.

The plots of the results from the Monte Carlo simulations of the constant turn rate (CTR) extended Kalman filter are presented in Fig D-1 to D-40. The results were characterized by nearly unbiased estimates of the x and y position coordinates in the FLIP frame, except during initial transients for estimates of the position and velocity. The small bias values observed for position were expected since the trajectories were well described as constant turn rate trajectories except during the intervals of rapid acceleration changes. Large mean errors occurred during these intervals and were a result of the inadequacy of the assumed constant turn rate model to represent the rapidly varying actual target acceleration. The mean velocity errors observed were much larger than expected and larger than the BM model. The cause of this was unknown and was ignored since the position was the critical performance parameter.

The mean errors of the position estimates for the CTR filter were within the envelope of plus or minus the square root of the filter's corresponding conditional variance, as expected, since the actual target acceleration was the same as the assumed model. However, the mean errors of the velocity estimates exhibited large biases, as previously mentioned, that exceeded the value of the filter's standard deviation. One method that might eliminate this problem would be to use adaptive Kalman filter techniques.

There were no plots generated for the adaptive multiple model extended Kalman filter. The multiple model filter was not able to select a 'best' filter from the bank of BM filters purely on the basis of the residuals. Instead the filter adaptively selected one filter and stayed with it throughout the simulated maneuver. The software seemed to work with no problems. The major problem was the multiple model filter's inability to select one BM filter over another because of the similarity of the residuals. Table 1 contains the statistics of the residuals taken from a simulated 20 g turn one second after the turn was initiated.

BM Filter	Mean	Standard Deviation
1	-4.7221	5.7016
2	-4.7353	5.6497
3	-4.7257	5.7227

Table 1. Residual Statistics

The mean of the residuals differed by less than one-half of one percent and the standard deviations differed by less than one percent. Thus at any given sample time, any one of the three filters could be the 'best' filter on the basis of it's residuals.

Comparison of the Three Filters

The figures of merit for each filter for the three trajectories are presented in Table 2 and Table 3. As seen from these tables, the constant turn rate filter provided estimates of position in the 10 g and 20 g maneuvers, where the time averaged scalar magnitudes of the mean error were significantly better than the Brownian motion estimates at high turn

rates. Also, the magnitude of standard deviation of error was slightly better in the CTR filter for both position and velocity in most cases.

For each dynamics model, a single filter, which tracks a 10 g turn the best, was used as a baseline against which to tune other filters. This filter was run against all three trajectories and then it's performance was the yardstick used to measure how well other filters were tuned. As it turned out, the CTR baseline model was the superior filter for the 2 g and 10 g case and was very close in performance to the selected 20 g CTR filter. The baseline BM filter also was the best tuned filter for two trajectories, the 20 g and 10 g case, and very close in performance to the selected 2 g filter. The BM 2 g filter experienced large oscillations after the maneuver started, the cause of which was not found.

The multiple model filter was a tenuous alternative at best, given the data contained in Table 2 and Table 3. Little performance improvement would have been gained if the filter would have worked as designed since the BM filters, used in the bank of filters, are so closely tuned. This performance improvement would have been prohibitively costly in terms of computer memory and run time. A better selection of filters for the multiple model algorithm would have been to use one BM filter and one CTR filter. A wider effective dynamic range, over which the multiple model filter could operate, would be gained by this set up. This set up was not explored since it wasn't realized until late in the research and the additional software that would be required would have needed much more time than available.

Another consideration in actual implementation would be the com-

Filter	Trajectory	Magnitude of the mean error (time averaged/Peak)			
		X Position (Pixels)	X Velocity (Pixels/sec)	Y Position (Pixels)	Y Velocity (Pixels/sec)
Brownian Motion	20 g Q=600	.326/.418	-.262/1.991	-.124/.443	-4.27/16.32
	10 g Q=600	.134/.278	.128/-1.492	-.031/.253	-2.08/8.22
	2 g Q=600	-.030/1.159	.177/7.74	-.011/-.124	-.422/1.801
	2 g Q=150	-.011/1.632	-1.489/8.830	-.007/.147	-.377/2.522
Constant Turn Rate	20 g Q=300	.100/.520	-1.764/-5.134	.045/.542	-7.67/18.04
	20 g Q=600	.034/.336	-2.412/-4.395	-.022/.368	-7.68/14.56
	10 g Q=300	.046/.349	-.473/-2.158	-.151/.292	-4.09/9.03
	2 g Q=300	.047/.310	.357/1.900	-.010/.110	-.826/1.947

Table 2. Comparison of the Mean Error Magnitudes

Filter	Trajectory	Magnitude of Standard Deviation of Error (time averaged actual/filter estimated)			
		X Position (Pixels)	X Velocity (Pixels/sec)	Y Position (Pixels)	Y Velocity (Pixels/sec)
Brownian Motion	20 g Q=600	.333/.189	2.304/1.766	.208/.189	1.513/1.766
	10 g Q=600	.368/.189	2.494/1.766	.181/.189	1.328/1.766
	2 g Q=600	.404/.189	2.733/1.766	.164/.189	1.255/1.766
	2 g Q=150	.378/.181	1.975/1.321	.154/.181	.926/1.321
Constant Turn Rate	20 g Q=300	.324/.185	2.086/1.654	.187/.185	1.346/1.654
	20 g Q=600	.323/.189	2.086/1.766	.181/.189	1.333/1.766
	10 g Q=300	.345/.185	2.187/1.654	.175/.185	1.217/1.654
	2 g Q=300	.358/.185	2.237/1.654	.162/.185	1.154/1.654

Table 3. Comparison of the Standard Deviations of the Errors

puter resources required. As mentioned above, the multiple model would require much more computation time and computer memory for the slight improvement in performance that might be gained if the filter worked. The CTR filter requires approximately the same computer time and memory as the BM filter. The savings in time for the state propagation in the CTR filter, from the Taylor series approximation, is offset by the requirement for the state transition matrix to be updated each sample period. The BM filter requires an 8x8 matrix multiplication each sample period for it's state propagation.

The total time required for the Monte Carlo simulation of the BM filter was about one-tenth of one percent faster than that of the CTR filter. This was unexpected since the CTR model is nonlinear and is due to the various approximations used in propagating the filter. The number of words of memory required were exactly the same for both filters as expected since both the BM and CTR filters have the same size and number of matrices.

VI Conclusions and Recommendations

Conclusions

The constant turn rate filter and the Brownian motion filter are both easily able to track the three trajectories evaluated for this thesis. When the target acceleration profiles are not demanding, as in the case for the 2 g pull-up, the performance is nearly the same for both filters. However, as the target acceleration profiles become more demanding (10 g and 20 g pull-up maneuvers) the performance of the CTR filter is substantially better than that of the BM filter. There is a tuning issue that requires further study to explore the reduction of the large excursions of the mean error outside of the filter standard deviation experienced during the periods of poorly modeled actual target accelerations in both filters. The computer resources required by the two simple extended Kalman filters were nearly identical. The total time required for the Monte Carlo simulation of the BM filter was about one-tenth of one percent faster than that of the CTR filter. The number of words of memory required were exactly the same for both filters.

The adaptive multiple model filter was found to be unsuitable for use in this environment. This was due to the BM filters being so closely matched in tuning that the residuals of one filter in the filter bank were not significantly different from the residuals of the other filters in the filter bank.

Recommendations

Both the Brownian motion and the constant turn rate filters need to undergo further testing of their robustness to variations in signal to noise ratio, measurement noise, tuning and imperfect initial conditions to provide additional insight into whether or not the CTR filter is truly the superior all purpose tracker. Slight modifications to the BM filter to turn it into a first order Gauss-Markov process, thus making the acceleration model more compatible with the actual target dynamics, should also be explored and compared against the CTR model.

Further testing of the adaptive multiple model filter also needs to be accomplished to verify the results of this report and to test the effects of having different filters included in the filter bank. A combination of the Brownian motion, the constant turn rate, and a Gauss-Markov filter would be one possibility. Also, update rate variations should be explored to study the effect this has on the performance of the filter. Another area that needs to be looked into is the possibility of using only the center 4x4 array of measurements and residuals would then be used to generate an 8x8 $\underline{A}(t_i)$ matrix. This should reduce or eliminate the scale factor employed in this effort.

Computer Support

The pace and extent of this research effort was significantly retarded by the computer support available at AFIT. An estimated 30 percent more work could have been accomplished with a more reliable and responsive system. The author encourages any steps to make AFIT's computer support better. Possible recommendations in-

clude more hardware support (i.e. it is unbelievable that AFIT has only one plotter), more personnel support, and AFIT's own dedicated computer. (Ref 3)

Bibliography

1. Department of Defense. "DoD High Energy Laser Program." Fact Sheet, July 1980.

2. Mercier, D.E., "An Extended Kalman Filter for Use in a Shared Aperature Medium Range Tracker." M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1978.

3. Jensen, R.L., and D.A. Harnly, "An Adaptive Distributed-Measurement Extended Kalman Filter for a Short Range Tracker." M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1979.

4. Maybeck, Peter S., Stochastic Models, Estimation, and Control, Volume 2. Unpublished text. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1978.

5. Maybeck, P.S., et al. "Robustness of a New Infrared Target Tracker." Proceedings of the 1980 IEEE National Aerospace and Electronics Conference, Dayton, Ohio, May 1980.

6. Ruoff, Tomas M., "Target Dynamics Model Alterations to a Short Range Kalman Filter," B.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, March 1981.

7. Maybeck, Peter S., Discussions concerning multiple model Kalman Filter implementation, October 1981.

APPENDIX A

"F" Matrix Derivation

The calculation of the linearized $\underline{F}$ matrix for the CTR filter was not as trivial as it was for the Brownian motion filter. The dynamics model for the CTR filter was nonlinear and therefore the point of linearization matters and it cannot be precomputed. The $\underline{F}$ matrix needed to be updated every sample period and serves as a quasi-static approximation of the $\underline{f}(x)$ over the sample period. Equation A-1 was used to calculate $\underline{F}$.

$$\underline{F}(t_i) = \left.\frac{\partial \underline{f}(\underline{x})}{\partial \underline{x}}\right|_{\underline{x}=\underline{\hat{x}}(t_i^+)} \tag{A-1}$$

where $\underline{f}(\underline{x})$ is as previously defined in equations 25 and 26.

The F matrix for the CTR model is in Equation (A-2).

$$\underline{F}(t_i) = \begin{bmatrix} 0 & 0 & F_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & F_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & F_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & F_4 & 0 & 0 \\ 0 & 0 & F_5 & F_6 & F_7 & F_8 & 0 & 0 \\ 0 & 0 & F_9 & F_{10} & F_{11} & F_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & F_{13} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & F_{14} \end{bmatrix} \tag{A-2}$$

where

$$F_1 = F_2 = F_3 = F_4 = 1$$

$$F_5 = -[2 \cdot x(3)(A_1 : A_2 \cdot x(6) - A_2^2 \cdot 2 \cdot x(3) + A_1 \cdot A_2^2]/A_1^3$$

$$F_6 = -2 \cdot x(3)[-A_1 \cdot A_2 \cdot x(5) - A_2^2 \cdot 2 \cdot x(4)]/A_1^3$$

$$F_7 = 2 \cdot x(3) \cdot x(4) \cdot A_2/A_1^2$$

$$F_8 = -2 \cdot x(3)^2 \cdot A_2/A_1^2$$

$$F_9 = -2 \cdot x(4) \cdot [A_1 \cdot A_2 \cdot x(6) - A_2^2 \cdot 2 \cdot x(3)]/A_1^3$$

$$F_{10} = -[2 \cdot x(4) \cdot (-A_1 \cdot A_2 \cdot x(5) - A_2^2 \cdot 2 \cdot x(4)) + A_1 \cdot A_2^2]/A_1^3$$

$$F_{11} = 2 \cdot A_2 \cdot x(4)^2/A_1^2$$

$$F_{12} = -2 \cdot A_2 \cdot x(4) \cdot x(3)/A_1^2$$

$$F_{13} = \exp(\Delta t/\tau)$$

$$F_{14} = F_{13}$$

where

$$A_1 = x(3)^2 + x(4)^2$$

$$A_2 = x(3) \cdot x(6) - x(4) \cdot x(5)$$

$\Delta t = 1/30$ seconds

τ = correlation time of atmospheric jitter

APPENDIX B

Main Program and Library Listings

This appendix contains the FORTRAN IV code for the CTR and multiple model algorithms presented earlier in the text. The program listings for the GM algorithm and the plotting routine can be found in Reference 2, Appendice F and G. The GM program can be obtained from the multiple model program by deleting two filters and the adaptive portions of the program.

Presented first is the CTR program listing, followed by the multiple model filter algorithm and its subroutine listings. Lastly, the library of subroutines common to all three main programs is given. One routine not shown is a matrix inversion routine from the IMSL library which is available at AFIT.

```
PROGRAM THESIS      74/74   OPT=2                    FTN 4.8+528

      PROGRAM THESIS(INPUT=/80,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE8)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
     *,ASPRO,FIMAX,UT(2,1),IMEAS,AR,SIGVF,PL2P(64,2),VMAX,SIGMFO
     *,RANGEO,RANGE,SIGFVF,COSTH,SNTH
      INTEGER UNI
      REAL IMAX
      DIMENSION Z(64),PHI(8,8),Q(3,3),
     * WORK(3,3),A(8),TEMP(8,8),TEMP1(8,8),SAVE(14),QFD1(8,8),
     *QD(8,8),PHIF(8,8),QFD(8,8),SQQD(8,8),XS(8),H(64,8),PFP(8,8),
     *XFP(8),PFM(8,8),HF(64),EXTRA(8,8),FIH(64,1),DXDXT(8,8),PPFP(8,8),
     *HT(8,64),WKAREA(50,50),C(5),R(64,64),RN(64,64),BD(8,2),PHIFT(8,8)
     *QFDMAX(8),HTZHO(2),DX(8),SIG(2),D(2),PFPOLD(8,8),XFPO(8)
     *,XFM(8)
      WRITE(6,1)
    1 FORMAT(1H1)
C
C        READ AND ECHO DATA
C
      READ *, SIGS1
      PRINT *, "RMS DYNAMICS FOR TRUTH MODEL,          SIGS1 = ",SIGS1
      READ *, SIGMAB
      PRINT *, "RMS TRUTH MODEL BACKGROUND NOISE,      SIGMAB = ",SIGMAB
      READ *, SIGFLR
      PRINT *, "RMS TRUTH MODEL FLIR NOISE,            SIGFLR = ",SIGFLR
      READ *, IMAX
      PRINT *, "TRUTH MODEL MAX INTENSITY,               IMAX = ",IMAX
      READ *, SIGAT
      PRINT *, "RMS ATMOSPHERICS FOR TRUTH MODEL,      SIGAT = ",SIGAT
      READ *, NRUN
      PRINT *, "NUMBER OF MONTE CARLO RUNS,              NRUN = ",NRUN
      READ *, TFINAL
      PRINT *, "FINAL TIME,                            TFINAL = ",TFINAL
      READ *,SIGMS
      PRINT *, "INITIAL RMS TRUTH MODEL SIGMA PERVEL, SIGMS = ",SIGMS
      READ *, ASPRO
      PRINT *, "TARGET ASPECT RATIO,                   ASPRO = ",ASPRO
      READ *,X0
      PRINT *, "INITIAL X POSITION,                      X0 = ",X0
      READ *,Y0
      PRINT *, "INITIAL Y POSITION,                      Y0 = ",Y0
      READ *,Z0
      PRINT *, "INITIAL Z POSITION,                      Z0 = ",Z0
      READ *,XDOT0
      PRINT *, "INITIAL VELOCITY, X DIRECTION ,        XDOT0 = ",XDOT0
      READ *,YDOT0
      PRINT *, "INITIAL VELOCITY, Y DIRECTION ,        YDOT0 = ",YDOT0
      READ *,ZDOT0
```

```
      PRINT *, "INITIAL VELOCITY, Z DIRECTION ,        ZDOT0 = ",ZDOT0
      READ *,ISPTL
      PRINT *, "SPATIAL NOISE: 1=YES, 0=NO             ISPTL = ",ISPTL
      IF(ISPTL.NE.1) GO TO 2
      READ *,C(1),C(2),C(3),C(4),C(5)
      PRINT *, "SPATIAL NOISE CORRELATION COEFFICIENTS: "
      PRINT *,C(1),C(2),C(3),C(4),C(5)
    2 CONTINUE
      PRINT *," "
      READ *, SIGMF0

      PRINT *, "INITIAL FILTER SIGMA VELOCITY,       SIGMF0 = ",SIGMF0
      READ *, AR0
      PRINT *, "INITIAL FILTER ASPECT RATIO,           AR0 = ",AR0
      READ *, SIGF2
      PRINT *, "RMS FILTER ATMOSPHERIC NOISE,         SIGF2 = ",SIGF2
      READ *, RF
      PRINT *, "FILTER MEASUREMENT NOISE RMS,           RF = ",RF
      READ *, SIGF10
      PRINT *, "INITIAL RMS DYNAMICS FOR FILTER,     SIGF10 = ",SIGF10
      READ *, FIMAX0
      PRINT *, "FILTER MAX INTENSITY,                FIMAX0 = ",FIMAX0
C
C        PARAMETER VALUES
C
C
C
C
C     **************************************************************
C     THE HIGH G TURNN IS PUT IN THIS VERSION
C     THE HIGH G TURNN IS PUT IN THIS VERSION
C     **************************************************************
C
C
C
C
C
C
      XFOV=8.
      YFOV=8.
      NPIX=8
      ATAU1=14.14
      ATAU2=659.5
      FTAU2=1./ATAU1
      DT = 1./30.
      IREF=1
      FIMAX=ABS(FIMAX0)
      SIGVF=ABS(SIGMF0)
      SIGF1=ABS(SIGF10)
```

```
      QFDMAX(1) = 2.
      QFDMAX(2)=QFDMAX(1)
      QFDMAX(3) = 14.
      QFDMAX(4)=QFDMAX(3)
      QFDMAX(5) = 20.
      QFDMAX(6)=QFDMAX(5)
      QFDMAX(7) = .5
      QFDMAX(8)=QFDMAX(7)

C        INITIALIZE TRUTH MODEL VARIABLES
C
      CALL RANSET(7562)
      ONE = 1
      NPS = 6
      NMS = NPIX**2
      NFS=8
      NFS2=NFS*NFS
      NFSM2=NFS-2
      NIS = 3
      RI=1./RF
      SN=SIGMAB/IMAX
      IF(SN.LE.0.) SN=.001
      SN=1./SN
      RANGE0=SQRT(X0**2+Y0**2+Z0**2)
      AGAIN = .351006534 * SIGAT
      IFILE=5
      DELT = -1.*DT
      DO 5 I=1,6
      BD(I,1)=0.
      BD(I,2)=0.
      DO 5 J=1,6
      QD(I,J) = 0.
      SQQD(I,J) = 0.
      PHI(I,J) = 0.
    5 CONTINUE
      FACT=(AGAIN**2)*(ATAU1**2)*(ATAU2**4)
      FACT=(AGAIN**2)*(ATAU1**2)*(ATAU2**4)
      FACT1 = ATAU1-ATAU2
      FACT2 = ATAU1+ATAU2
      FACT3 = 2.*ATAU2
      G1 = FACT/(FACT1**4)
      G2 = FACT/(FACT1**3)
      G3 = FACT/(FACT1**2)
      R1 =  1.- EXP(2.*ATAU1*DELT)
      R2 =  1.- EXP(FACT2*DELT)
      R3 =  1.- EXP(2.*ATAU2*DELT)
      R4 = DT*EXP(DELT*FACT2)
      R5 = DT*EXP(2.*ATAU2*DELT)
C
C        FILL OUT TRUTH MODEL PHI MATRIX.
C        SEE MERCIER'S THESIS FOR DERIVATION
C
```

```
      PHI(1,1)= 1.
      PHI(2,2) = PHI(1,1)
      PHI(3,3) = EXP(ATAU1*DELT)
      PHI(4,4) = EXP(ATAU2*DELT)
      PHI(4,5) = DELT*PHI(4,4)
      PHI(5,5) = PHI(4,4)
      PHI(6,6) = PHI(3,3)
      PHI(7,7) = PHI(4,4)
      PHI(7,8) = PHI(4,5)
      PHI(8,8) = PHI(5,5)
      WRITE(6,11)
   11 FORMAT(///2X,"THE TRUTH MODEL STATE TRANSITION MATRIX IS:"/)
C
C        FILL OUT DISCRETE INPUT MATRIX
C
      BD(1,1)= DT
      BD(2,2)= DT
      WRITE(6,15)
   15 FORMAT(///2X,"THE TRUTH MODEL INPUT MATRIX IS:"/)
C
C        FILL THE QD MATRIX WITH VALUES USING EXACT INTEGRATION
C        SEE MERCIER'S THESIS FOR DERIVATION
C
      QD(1,1)= SIGS1
      QD(2,2) = QD(1,1)

      QD(3,3) = (G1*R1)/(2.*ATAU1)
      QD(3,4) = R2*(G2/FACT2**2-G1/FACT2)-R4*G2/FACT2
      QD(3,5) = G2*R2/FACT2
      QD(4,3) = QD(3,4)
      QD(4,4) = R3*(G1/FACT3-2.*G2/FACT3**2+2.*G3/FACT3**3)-
     * R5*(G2/ATAU2+G3*DT/FACT3-2.*G3/FACT3**2)
      QD(4,5) = R3*(G3/FACT3**2-G2/FACT3)-R5*G3/FACT3
      QD(5,3) = QD(3,5)
      QD(5,4) = QD(4,5)
      QD(5,5) = R3*G3/FACT3
      DO 20 I=3,5
      DO 20 J=3,5
      QD(I+3,J+3) = QD(I,J)
      Q(I-2,J-2) = QD(I,J)
   20 CONTINUE
      WRITE(6,30)
   30 FORMAT(///2X,"THE TRUTH MODEL QD MATRIX IS:"/)
C
C        TAKING CHOLESKY SQUARE ROOT OF QD
C
      SQQD(1,1) = SQRT(QD(1,1))
      SQQD(2,2) = SQQD(1,1)
      CALL CHOLESK(Q,WORK,NIS)
      DO 33 I=1,NIS
      DO 33 J=1,NIS
      SQQD(I+2,J+2) = WORK(J,I)
      SQQD(I+5,J+5) = WORK(J,I)
   33 CONTINUE
```

```
      WRITE(6,35)
   35 FORMAT(///2X,"THE CHOLESKY SQUARE ROOT OF QD IS:"/)
      CALL MOUT(SQQD,NPS,NPS)
      IF (ISPTL.NE.1) GO TO 41
C
C        SET UP SPATIAL NOISE CORRELATION COEFFICIENT MATRIX
C
      N=64
      M=8
      DO 36 I=1,N
      R(I,I)=1.
      IF (I.GE.64) GO TO 36
      R(I,I+1)=C(1)
      IF (I.GE.63) GO TO 36
      R(I,I+2)=C(3)
      IF (I.GE.57) GO TO 36
      R(I,I+5)=C(4)
      R(I,I+7)=C(2)
      R(I,I+8)=C(1)
      R(I,I+9)=C(2)
      R(I,I+10)=C(4)
      IF (I.GE.49) GO TO 36
      R(I,I+14)=C(5)
      R(I,I+15)=C(4)
      R(I,I+16)=C(3)
      R(I,I+17)=C(4)
      R(I,I+18)=C(5)
   36 CONTINUE
      DO 37 I=1,M
      R(8*I-7,8*I)=0.0
      R(8*I-7,8*I-1)=0.0
      R(8*I-6,8*I)=0.0
      IF (I.GE.8) GO TO 37
      R(8*I,8*I+1)=0.0
      R(8*I,8*I+2)=0.0
      R(8*I-1,8*I+1)=0.0
      R(8*I-7,8*I+7)=0.0
      R(8*I-7,8*I+8)=0.0
      R(8*I-6,8*I+8)=0.0
      IF (I.GE.7) GO TO 37
      R(8*I,8*I+9)=0.0
      R(8*I,8*I+10)=0.0
      R(8*I-1,8*I+9)=0.0
      IF (I.GE.6) GO TO 37
      R(8*I,8*I+17)=0.0
      R(8*I,8*I+18)=0.0
      R(8*I-1,8*I+17)=0.0
   37 CONTINUE
      DO 38 I=1,N
      L=I+1
      DO 38 J=L,N
      R(J,I)=R(I,J)
   38 CONTINUE
      DO 39 I=1,N
```

```
      DO 39 J=1,N
      RN(I,J)=SIGMA**R(I,J)
   39 CONTINUE
      WRITE(6,44)
   44 FORMAT(///2X,"64 X 64 SPATIAL CORRELATION MATRIX:"/)
C
C        COMPUTE THE CHOLESKY SQUARE ROOT OF RN
C
      CALL CHOLESK(RN,R,64)
      WRITE(6,46)
   46 FORMAT(///2X,"THE CHOLESKY SQUARE ROOT OF RN:"/)
      GO TO 42
   41 DO 43 I=1,64
   43 R(I,I)=1.
   42 CONTINUE
C
C        SET UP FILTER MATRICES.
C
      DO 51 I=1,NFS
      DO 51 J=1,NFS
      PHIF(I,J)=0.
   51 QFD(I,J)=0.
C
C        FILL OUT FILTER PHI MATRIX
C
C
C       IT IS FROM HERE THAT THE ORIGINAL PHIF ELEMENTS WHERE REMOVED
C
      DO 55 I=1,NFS
      DO 55 J=1,NFS
   56 PHIFT(I,J)=PHIF(J,I)
C

C        FILL OUT FILTER DISCRETE QFD MATRIX
C        FOR START OF ACQUISITION PHASE
C
      QFD(5,5)=SIGF1*DT
      QFD(5,6)=QFD(5,5)
      QFD (7,7) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD (8,8)=QFD (7,7)
      WRITE(6,40)
   40 FORMAT(///2X,"THE FILTER STATE TRANSITION MATRIX IS:"/)
      CALL MOUT(PHIF,NFS,NFS)
      WRITE(6,45)
   45 FORMAT(///2X,"THE INITIAL FILTER QD MATRIX IS:"/)
      PRINT *," "
      PRINT *," "
C
      PRINT *,"          BEGIN THE MONTE CARLO SIMULATION "
C
      DO 99 L=1,NRUN
      TIME = 0.
      XCENTR=0.
      YCENTR=0.
```

```
      FIMIN =0.
      IMEAS=0
      FIMAX=ABS(FIMAX0)
      AR=AR0
      SIGVF=ABS(SIGMF0)
      SIGF1=ABS(SIGF10)
      VARYQ=ABS(SIGF10)
      TRXXT=200.
      MANIND=0
C
C        RESET INITIAL CONDITIONS FOR NEW RUN
C
      DO 46 I=1,NFS
      XS(I) = 0.
   46 CONTINUE
      DO 47 I=1,NFS
      XFP(I) = 0.
      XFM(I)=0.
      DO 47 J=1,NFS
      QFD(I,J)=0.
      PFP(I,J) = 0.
   47 CONTINUE
      XRES1 = 0.
      XRES2 = 0.
      XRES3 = 0.
      XRES4 = 0.
      XRES5 = 0.
      XRES6 = 0.
      XRES6 = 0.
C
C        PROVIDE FILTER WITH INITIAL VELOCITY AND POSITION
C
      RHOR=(X0**2+Z0**2)
      RANGE=(RHOR+Y0**2)
      XFP(3)=(Z0*XDOT0-X0*ZDOT0)/(RHOR*.00002)
      RHOR=SQRT(RHOR)

      XFP(4)=(RHOR*YDOT0-Y0*((X0*XDOT0+Z0*ZDOT0)/RHOR))/(RANGE*.00002
      PRINT *," RUN NUMBER ",NRUN
      PRINT *," RHOR=",RHOR," RANGE=",RANGE," XFP(3)=",XFP(3),
     *" XFP(4)=",XFP(4)
C
C
C     THESE ARE THE DYNAMIC MODEL CHANGES MADE
C
      Q1=XFP(1)
      Q2=XFP(2)
      Q3=XFP(3)
      Q4=XFP(4)
      Q4=XFP(4)
      Q5=XFP(5)
      Q6=XFP(6)
      Q7=XFP(7)
      Q8=XFP(6)
```

```
C
C        PERFORM TRUTH MODEL SIMULATION
C
      VTIME=TIME-DT/2.
      IF(TIME-2.0) 500,505,505
  500 XDOT=-1000.
      YDOT=0.
      ZDOT=0.
      XVEH=X0+XDOT*TIME
      YVEH=Y0
      ZVEH=Z0
       GO TO 510
C
C
C
C
CTHIS IS A 2 G TURN
C
C
C
  505 XDOT=-1000.*COS(.0196*(VTIME-2.))
      YDOT=1000.*SIN(.0196*(VTIME-2.))
      XVEH=X0-2000.-51020.*SIN(.0196*(TIME-2.))
      YVEH=Y0+51020.*(1.-COS(.0196*(TIME-2.)))
      ZVEH=Z0
  510 CONTINUE
      RHOR=(XVEH**2+ZVEH**2)
      RANGE=(RHOR+YVEH**2)
      UT(1,1)=(ZVEH*XDOT-XVEH*ZDOT)/(RHOR*.00002)
      RHOR=SQRT(RHOR)
      UT(2,1)=(RHOR*YDOT-YVEH*((XVEH*XDOT+ZVEH*ZDOT)/RHOR))/(RANGE
     -*.00002)
      RANGE=SQRT(RANGE)
      VMAX=SQRT(XDOT**2+YDOT**2+ZDOT**2)/(RANGE*.00002)
      XCENTR=XCENTR+UT(1,1)*BD(1,1)
      YCENTR=YCENTR+UT(2,1)*BD(2,2)
      CALL NOISE (NPS,W)
      CALL MMPY(TEMP,SQQD,W,NPS,NPS,ONE)
      CALL MMPY(TEMP1,PHI,XS,NPS,NPS,ONE)
      CALL MADD(XS,TEMP,TEMP1,NPS,ONE,ONE)
      CALL MMPY(TEMP1,BD,UT,NPS,2,ONE)
      CALL MADD(XS,XS,TEMP1,NPS,ONE,ONE)
      CALL SHIFTA(XFP,XFPO,NFS,NFS)
C
C        FILTER STATE PROPAGATION.

      OMEG=(Q3*Q6-Q4*Q5)/Q10
       XFM(1)=XFP(1)+Q3*DT
       XFM(2)=XFP(2)+Q4*DT
       XFM(3)=XFP(3)+Q5*DT
       XFM(4)=XFP(4)+Q6*DT
      XFM(5)=XFP(5)+(-OMEG**2)*Q3*DT
       XFM(6)=XFP(6)+(-OMEG**2)*Q4*DT
      XFM(7)=XFP(7)-(Q7/FTAU2)*DT
      XFM(8)=XFP(8)-(Q8/FTAU2)*DT
```

```
C
C        FORM CENTROID POSITION AND FILL TRUTH ARRAY
C
      XPEAK = XS(1) + XS(3) + XS(4)  -XFM(1)
      YPEAK = XS(2) + XS(6) + XS(7)  -XFM(2)
      IF(ABS(XPEAK).GT.3.*ASPRO*SIGMS) GO TO 101
      IF(ABS(YPEAK).GT.3.*ASPRO*SIGMS) GO TO 102
      CALL MEA (XPEAK,YPEAK,4,Z,R)
      IF(TIME.EQ.0) GO TO 1002
      PRINT *," "
      PRINT *,"Z"
      CALL MOUT(Z,NPIX,NPIX)
 1002 CONTINUE
C
C        SEARCH FOR FIMAX AND FIMIN
C        SEARCH FOR FIMAX AND FIMIN
C
      IF(FIMAXO.GT.0.) GO TO 59
      FIMXO=FIMAX
      FIMAX=0.
      FIMIN=0.
      DO 55 I=1,NMS
      IF(Z(I).GT.FIMAX) FIMAX = Z(I)
   55 IF(Z(I).LT.FIMIN) FIMIN = Z(I)
      FIMAX=FIMAX+.14*SIGVF**2-1.52*SIGVF+4.35
      FIMAX= .8*FIMXO+ .2*FIMAX
   59 CONTINUE
C
C        FILTER COVARIANCE PROPAGATION
C
      CALL SHIFTA(PFP,PFPOLD,NFS2,NFS2)
      CALL MMPY(EXTRA,PHIF,PFP,NFS,NFS,NFS)
      CALL MMPY(PPFP,EXTRA,PHIFT,NFS,NFS,NFS)
C        PPFP= PHI * P(TI-1) + * PHIT
      CALL MADD(PFM,PPFP,QFD,NFS,NFS,ONE)
C        PFM = P(TI-)=PHI * P(TI-1)+ * PHIT + G * Q * GT
C
C        PERFORM MEASUREMENT UPDATE FOR THE FILTER
C        INVERSE COVARIANCE FORM
C        FORM FILTER CENTROID POSITION AND FILL OUT NON LINEAR
C        SMALL H. CALCULATE PARTIAL SMALL H PARTIAL X
C
      XPEAK = XFM(7)
      YPEAK = XFM(8)
      IF((XFM(3)**2+XFM(4)**2).EQ.0.) XFM(3)=.001
      CALL MEASF(XPEAK,YPEAK,ONE,HF,H,XFM)
```

```
      DO 60 I=1,NMS
      DO 60 J=1,NFS
      HT(J,I) = H(I,J)
   60 CONTINUE
      CALL MMPY(PFP,HT,H,NFS,NMS,NFS)
C        PFP= HT * H  (R-1 IS SCALAR AND MULTIPLIED LATER)
      IDGT = 0
      CALL LIN2F(PFP,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
C        EXTRA  P(TI-)-1
      DO 65 I=1,NFS
      DO 65 J=1,NFS
      PFP(I,J)=PFP(I,J)*RI + EXTRA(I,J)
   65 CONTINUE
C        PFP= P(TI+)-1= HT * R-1 * H + P(TI-)-1
      IDGT = 0
      CALL LIN2F(PFP,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
      DO 70 I=1,NFS
      DO 70 J=1,NFS
      PFP(I,J) = EXTRA(I,J)
   70 CONTINUE
      DO 130 I=1,NFS
      IF(PFP(I,I).GT.0.) GO TO 130
      PRINT *,"PFP(",I,I,")=",PFP(I,I)
      PFP(I,I)=PFPOLD(I,I)
  130 CONTINUE
C        PFP=P(TI+)
      CALL MADD(RIH,Z,HF,NMS,ONE,NFS)
C        RIH=RESIDUALS= Z-SMALL H
      IF(SIGMF0.GT.0.) GO TO 62
C
C        COMPUTE NEW ESTIMATE OF SIGVF AND AR
C
      DO 58 I=1,NMS
      AR= AR+.001*(RIH(I)*PL2P(I,1))
      SIGVF=SIGVF+.001*(RIH(I)*PL2P(I,2))
   58 CONTINUE
      IF(AR.LE.1.) AR=1.
      IF(SIGVF.LE.0.) SIGVF=SIGMF0
   62 CONTINUE
      CALL MMPY(EXTRA,HT,RIH,NFS,NMS,ONE)
C        EXTRA= HT * (Z-SMALL H)
      CALL MMPY(DX,PFP,EXTRA,NFS,NFS,ONE)
C        DX = P(TI+) * HT * (Z - SMALL H)
      DO 75 I=1,NFS
      DX (I)=DX (I)*RI
   75 CONTINUE
C        DX =DELTA X = P(TI+) * HT * R-1 * (Z - SMALL H)
      DO 76 I=1,NFS
      DO 76 J=1,NFS
      DXDXT(I,J)=DX (I)*DX (J)
   76 CONTINUE
      IF(TIME .LT. 1.95) GO TO 1005
 1005 CONTINUE
```

```
      DXDXT  (X(TI+) - X(TI-)) * (X(TI+) - X(TI-))T
      TRXXTO=TRXXT
      TRXXT=0.
      DO 79 I=1,NFS

79    TRXXT=TRXXT+DXDXT(I,I)
      THRESH = ABS(EXTRA(1,1)) + ABS(EXTRA(2,1))
      IF (THRESH.LE.2000.) GO TO 112
      IF(MANIND.EQ.1) GO TO 112
      ECC=SQRT(1.-(1./AR**2))
      SIG(1)=SQRT(SIGPVF**2/(1.-(ECC*COSTH)**2))
      SIG(2)=SQRT(SIGPVF**2/(1.-(ECC*SINTH)**2))
      PRINT *," SIG(1)=",SIG(1),"  SIG(2)=",SIG(2),"ECC=",ECC
      DO 116 I=1,2
      DHTZ=EXTRA(I)
      ALOD=(ALOG10(ABS(DHTZ))+.4651*SIG(I)-3.9)/.82
      D(I)=EXP(2.303*ALOD)
      ADHTZ = ABS(DHTZ)
      IF (ADHTZ.GT.100.) GO TO 791
      D(I) = 0.
791   CONTINUE
      PRINT *,"D(",I,")=", D(I),"  DHTZ= ",DHTZ
      XFP(I+4)=(2.*D(I)/DT**2)*DHTZ/ABS(DHTZ)+XFPO(I+4)
116   CONTINUE
      DO 114 I=1,NFS
      IF(I.EQ.5) GO TO 114
      IF(I.EQ.6) GO TO 114
      XFP(I)=XFPO(I)
114   CONTINUE
      CALL SHIFTA(PFPOLD,PFP,NFS2,NFS2)
      DO 140 I=3,4
      VFACTOR=10./SQRT(PFP(I,I))
      AFACTOR=300./SQRT(PFP(I+2,I+2))
      DO 139 J=1,NFS
      PFP(I,J)=PFP(I,J)*VFACTOR
      PFP(J,I)=PFP(J,I)*VFACTOR
      PFP(I+2,J)=PFP(I+2,J)*AFACTOR
      PFP(J,I+2)=PFP(J,I+2)*AFACTOR
139   CONTINUE
140   CONTINUE
      MANIND=1
      GO TO 121
112   CONTINUE
      MANIND=0
      TRXXT=.8*TRXXTO+.2*TRXXT
      CALL MADD(XFP,DX,XFM,NFS,ONE,ONE)
C        XFP=X(TI+)= P(TI+) * R-1 * HT * (Z - SMALL H) + X(TI-)
      IF (TIME.LT.0.04) GO TO 202
      XRES1 = XFP(1) - XFM(1)
      XRES2 = XFP(2) - XFM(2)
      XRES3 = XFP(3) - XFM(3)
      XRES4 = XFP(4) - XFM(4)
      XRES5 = XFP(5) - XFM(5)
      XRES6 = XFP(6) - XFM(6)
```

```
      AXR1 = ABS(XRES1)
      AXR2 = ABS(XRES2)
      IF (AXR1.GT.0.5) GO TO 201
      XRES1 = 0.
      XRES3 = 0.
      XRES5 = 0.
  201 CONTINUE
      IF (AXR2.GT.0.5) GO TO 202

      XRES2 = 0.
      XRES4 = 0.
      XRES6 = 0.
  202 CONTINUE
      IF(TIME .LT. 1.95) GO TO 1006
 1006 CONTINUE
C
C     THIS IS THE NEW DYNAMICS MODEL IN THE TIME LOOP
C
      Q1=XFP(1)
      Q2=XFP(2)
      Q3=XFP(3)
      Q4=XFP(4)
      Q5=XFP(5)
      Q6=XFP(6)
      Q7=XFP(7)
      Q8=XFP(8)
      Q10=Q3**2+Q4**2
      PHIF(1,1)=1
      PHIF(2,2)=1
      PHIF(2,2)=1
      PHIF(3,3)=1
      PHIF(4,4)=1
      PHIF(1,3)=DT
      PHIF(2,4)=DT
      PHIF(3,5)=DT
      PHIF(4,6)=DT
      A1SQ=A1**2
      A1CUBE=A1**3
      A2SQ=A2**2
      PHIF(5,3)=-(2*Q3*(A12*Q6-A2SQ*2*Q3) +A2SQ*A1)*DT/A1CUBE
      PHIF(5,4)=-2*Q3*(-A12*Q5-A2SQ*2*Q4)*DT/A1CUBE
      PHIF(5,5)=1+2*Q3*Q4*A2*DT/A1SQ
      PHIF(5,6)=-2*Q3**2*A2*DT/A1SQ
      PHIF(6,3)=-2*Q4*DT*(A12*Q6-A2SQ*2*Q3)/A1CUBE
      PHIF(6,4)=-(2*Q4*(-A12*Q5-A2SQ*2*Q4)+A2SQ*A1)*DT/A1CUBE
      PHIF(6,5)=2*A2*Q4**2*DT/A1SQ
      PHIF(6,6)=1-2*A2*Q4*Q3*DT/A1SQ
      PHIF(7,7)=EXP(DELT/FTAU2)
      PHIF(8,8)=PHIF(7,7)
      DO 5011 J=1,NFS
      DO 5011 I=1,NFS
 5011 PHIFT(I,J)=PHIF(J,I)
      OMEGA=(Q3*Q6-Q4*Q5)/Q10**.5
```

```
C
C        ACQUISITION OR ESTIMATION OF QFD
C
      IF(SIGF1..GT.0.) GO TO 74
      CALL SHIFTA(QFD,QFD1,NFS2,NFS2)
      IF(TIME-.1) 72,72,57
   57 IF(TIME-.5) 61,60,60
   61 IF(TRXXT-1(..) 72,77,77
C
C        ACQUISITION DOUBLE CHANGE OF QFD
C
   72 VARYQ=VARYQ-SIGF1*.1E+44+++++
      QFD(1,1)=DT**5*VARYQ/20.
      QFD(1,3)=DT**4*VARYQ/8.
      QFD(1,5)=DT**3*VARYQ/6.
      QFD(2,2)=QFD(1,1)
      QFD(2,4)=QFD(1,3)
      QFD(2,6)=QFD(1,5)
      QFD(3,1)=QFD(1,3)
      QFD(3,3)=DT**3*VARYQ/3.
      QFD(3,5)=DT**2*VARYQ/2.
      QFD(4,2)=QFD(3,1)
      QFD(4,4)=QFD(3,3)
      QFD(4,4)=QFD(3,3)
      QFD(4,6)=QFD(3,5)
      QFD(5,1)=QFD(1,5)
      QFD(5,3)=QFD(3,5)
      QFD(5,5)=VARYQ*DT
      QFD(6,2)=QFD(1,5)
      QFD(6,4)=QFD(3,5)
      QFD(6,6)=QFD(5,5)
      QFD (7,7) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD (8,8)=QFD (7,7)
      GO TO 74
C
C        ESTIMATION OF QFD
C
   77 PRINT *," "
      PRINT *," ADAPTION STARTED AT ",TIME," SEC."
   60 CALL MADD(QFD,DXDXT,PFP,NFS,NFS,ONE)
      CALL MADD(QFD,QFD,PPFP,NFS,NFS,-1)
C        QFD= DXDXT + P(TI+) - PHI * P(TI-1)+ * PHIT
C
C        BOUNDING QFD
C
      DO 84 I=1,NFS
      QFACTOR=1.
      IF(QFD(I,I).GT.0.) GO TO 86
      QFACTOR=0.
      QFD(I,I) = 0.1
      GO TO 85
   86 CONTINUE
```

```
      IF(SQRT(QFD(I,I)).GT.QFDMAX(I)) QFACTOR=QFDMAX(I)/SQRT(QFD(I,I))
      IF(QFACTOR.GE.1.) GO TO 64
      QFD(I,I)=QFACTOR**2*QFD(I,I)
   63 CONTINUE
      DO 62 J=1,NFS
      IF(I.EQ.J) GO TO 62
      QFD(I,J)=QFD(I,J)*QFACTOR
   62 CONTINUE
      DO 63 K=1,NFS
      IF(I.EQ.K) GO TO 63
      QFD(K,I)=QFD(K,I)*QFACTOR
   63 CONTINUE
   64 CONTINUE
      DO 73 I=1,NFS
      DO 73 J=1,NFS
   73 QFD(I,J)= .2*QFD(I,J)+.8*QFD1(I,J)
   74 CONTINUE
C
C
C
C        WRITE DATA TO FILE TAPE8
C
      SAVE(1) = XS(1)
      SAVE(2)   UT(1,1)
      SAVE(3) = XS(2)
      SAVE(4) = UT(2,1)
      SAVE(5) = XFP(1)
      SAVE(6) = XFP(3)
      SAVE(7) = XFP(2)
      SAVE(8) = XFP(4)
      SAVE(9) = XCENTR
      SAVE(10) = YCENTR
      SAVE(11) = PFP(1,1)
      SAVE(12) = PFP(3,3)
      SAVE(13) = PFP(2,2)
      SAVE(14) = PFP(4,4)
      WRITE(8) SAVE
      GO TO 50
  101 PRINT *, "LOST TRACK, X CHANNEL, MEAS CALLED ",IMEAS," TIMES.
     *RUN ",L
      PRINT *,"  TIME = ",TIME
      GO TO 105
  102 PRINT *, "LOST TRACK, Y CHANNEL, MEAS CALLED ",IMEAS," TIMES.
     *RUN ",L
      PRINT *,"  TIME = ",TIME
  105 DO 110 J=1,14
  110 SAVE(J)=0.
  106 WRITE(8) SAVE
      TIME=TIME+DT
      IF(TIME.GT.TFINAL) GO TO 99
      GO TO 106
   99 CONTINUE
      STOP "FINISH"
      END
```

```
      PROGRAM THESIS(INPUT=/61,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,T
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
     *,ASPRO,FIMAX,UT(2,1),IMEAS,AR,SIGVF,PL2P(64,2),YMAX,SIGMFO
     *,RANGEO,RANGE,SIGPVF,CSTH,SNTH
      INTEGER ONE
      REAL IMAX
      DIMENSION Z(64),PHI(8,8),Q(3,3),
     * WORK(3,3),W(8),TEMP(8,8),TEMP1(8,8),SAVE(14),QFD1(8,8),
     *QD(8,8),PHIF(8,8),QFD(8,8),SQD(8,8),XS(8),H(64,8),PFP(8,8),
     *XFP(8),PFM(8,8),HF(64),EXTRA(8,8),RIH(64,1),DXDXT(8,8),PPF(8,8),
     *HT(8,64),MEARE(50,50),G(5),R(64,64),FM(64,64),ED(8,2),PHIFT(8,8)
     *HTZHO(2),DX(8) SIG(2),D(2),PFPOLD(8,8),XFPO(8),XFP1(8),
     *XFP2(8),XFP3(8),XFFS1(8),XFFS2(8),XFFS3(8),XFM1(8),XFM2(8),
     *XFM3(8),HT1(8,64),HT2(8,64),HT3(8,64)
     *,H1(64,8),H2(64,8),H3(64,8)
     *,XFM(8),PFP1(8,8),PFP2(8,8),PFP3(8,8),PFM1(8,8),PFM2(8,8),PFM3(8,
     *),PHIF1(8,8),PHIF2(8,8),PHIF3(8,8),PHIFT1(8,8),PHIFT2(8,8),
     *PHIFT3(8,8),QFD1(8,8),QFD2(8,8),QFD3(8,8),HF1(64),HF2(64),HF3(64),
     *RIH1(64),RIH2(64),RIH3(64),EXTRA1(8,8),EXTRA2(8,8),EXTRA3(8,8)
     *,EXTP1(8),EXTR2(8),EXTR3(8)
      WRITE(6,1)
    1 FORMAT(1H1)
C
C        READ AND ECHO DATA
C
      READ *, SIGS1
      PRINT *, "RMS DYNAMICS FOR TRUTH MODEL.          SIGS1 = ",SIGS1
      READ *, SIGMAB
      PRINT *, "RMS TRUTH MODEL BACKGROUND NOISE.      SIGMAB = ",SIGMAB
      READ *, SIGFLR
      PRINT *, "RMS TRUTH MODEL FLIR NOISE.            SIGFLR = ",SIGFLR
      READ *, IMAX
      PRINT *, "TRUTH MODEL MAX INTENSITY,             IMAX = ",IMAX
      READ *, SIGAT
      PRINT *, "RMS ATMOSPHERICS FOR TRUTH MODEL,      SIGAT = ",SIGAT
      READ *, NRUN
      PRINT *, "NUMBER OF MONTE CARLO RUNS.            NRUN = ",NRUN
      READ *, TFINAL
      PRINT *, "FINAL TIME,                            TFINAL = ",TFINAL
      READ *,SIGMS
      PRINT *, "INITIAL RMS TRUTH MODEL SIGMA PERVEL, SIGMS = ",SIGMS
      READ *, ASPRO
      PRINT *, "TARGET ASPECT RATIO,                   ASPRO = ",ASPRO
      READ *,X0
      PRINT *, "INITIAL X POSITION,                    X0 = ",X0
      READ *,Y0
      PRINT *, "INITIAL Y POSITION,                    Y0 = ",Y0
      READ *,Z0
      PRINT *, "INITIAL Z POSITION,                    Z0 = ",Z0
      READ *,XDOT0
      PRINT *, "INITIAL VELOCITY, X DIRECTION ,        XDOT0 = ",XDOT0
      READ *,YDOT0
      PRINT *, "INITIAL VELOCITY, Y DIRECTION ,        YDOT0 = ",YDOT0
      READ *,ZDOT0
      PRINT *, "INITIAL VELOCITY, Z DIRECTION ,        ZDOT0 = ",ZDOT0
```

```
      ) *,ISPTL
      PRINT *, "SPATIAL NOISE 1-YES. 0-NO                    ISPTL = ",ISPTL
      IF(ISPTL.NE.1) GO TO 2
      READ *,C(1),C(2),C(3),C(4),C(5)
      PRINT *, "SPATIAL NOISE CORRELETION COLFFICIENTS "
      PRINT *,C(1),C(2),C(3),C(4),C(5)
    2 CONTINUE
      PRINT *," "
      READ *, SIGMFG
      PRINT *, "INITIAL FILTER SIGMA VELOCITY,       SIGMFG = ",SIGMFG
      READ *, ARC
      PRINT *, "INITIAL FILTER ASPECT RATIO,           ARC = ",ARC
      READ *, SIGF2
      PRINT *, "RMS FILTER ATMOSPHERIC NOISE,        SIGF2 = ",SIGF2
      READ *, RF
      PRINT *, "FILTER MEASUREMENT NOISE RMS.          RF = ",RF
      READ *, SIGF1C
      PRINT *, "INITIAL RMS DYNAMICS FOR FILTER,     SIGF1C = ",SIGF1C
      READ *, FIMAXC
      PRINT *, "FILTER MAX INTENSITY,                FIMAXC = ",FIMAXC
C
C        PARAMETER VALUES
C
C
C
C     ***************************************************************
C     THE HIGH G TURNN IS PUT IN THIS VERSION
C     THE HIGH G TURNN IS PUT IN THIS VERSION
C     ***************************************************************
C
C
C
C
      XFOV=8.
      YFOV=8.
      NPIX=8
      ATAU1=1+.1L
      ATAU2=659.3
      FTAU2=1./ATAU1
      DT = 1./30.
      IREF=1
      SIGVF=ABS(SIGMFG)
      FIMAX=ABS(FIMAXC)
C
C        INITIALIZE TRUTH MODEL VARI-BLES
C
      CALL RANSET(75682)
      ONE = 1
      NPS = 8
      NMS = NPIX**2
      NFS=8
      NFS2=NFS*NFS
      NFSM2=NFS-2
      NIS = 3
      RI=1./RF
      SN=SIGMAB/IMAX
      IF(SN.LE.0.) SN=.001
```

```
      SN=1./SN
      RANGE0=SQRT(X0**2+Y0**2+Z0**2)
      AGAIN = .351006534 * SIGAT
      IFILE=6
      DELT = -1.*DT
      DO 5 I=1,8
      BD(I,1)=0.
      BD(I,2)=0.
      DO 5 J=1,8
      QD(I,J) = 0.
      SQQD(I,J) = 0.
      PHI(I,J) = 0.
    5 CONTINUE
      FACT=(AGAIN**2)*(ATAU1**2)*(ATAU2**4)
      FACT=(AGAIN**2)*(ATAU1**2)*(ATAU2**4)
      FACT1 = ATAU1-ATAU2
      FACT2 = ATAU1+ATAU2
      FACT3 = 2.*ATAU2
      G1 = FACT/(FACT1**4)
      G2 = FACT/(FACT1**3)
      G3 = FACT/(FACT1**2)
      R1 =  1.- EXP(2.*ATAU1*DELT)
      R2 =  1.- EXP(FACT2*DELT)
      R3 =  1.- EXP(2.*ATAU2*DELT)
      R4 = DT*EXP(DELT*FACT2)
      R5 = DT*EXP(2.*ATAU2*DELT)
C
C        FILL OUT TRUTH MODEL PHI MATRIX.
C        SEE MERCIER'S THESIS FOR DERIVATION
C
      PHI(1,1)= 1.
      PHI(2,2) = PHI(1,1)
      PHI(3,3) = EXP(ATAU1*DELT)
      PHI(4,4) = EXP(ATAU2*DELT)
      PHI(4,5)=DT*PHI(4,4)
      PHI(5,5) = PHI(4,4)
      PHI(6,6) = PHI(3,3)
      PHI(7,7) = PHI(4,4)
      PHI(7,8) = PHI(4,5)
      PHI(8,8) = PHI(5,5)
      WRITE(6,11)
   11 FORMAT(///2X,"THE TRUTH MODEL STATE TRANSITION MATRIX IS:"/)
C
C        FILL OUT DISCRETE INPUT MATRIX
C
      BD(1,1)= DT
      BD(2,2)= DT
      WRITE(6,15)
   15 FORMAT(///2X,"THE TRUTH MODEL INPUT MATRIX IS:"/)
C
C        FILL THE QD MATRIX WITH VALUES USING EXACT INTEGRATION
C        SEE MERCIER'S THESIS FOR DERIVATION
C
```

```
      Q10=Q3**2+Q4**2
      PHIF(4,6)=DT
      PHIF(3,5)=DT
      PHIF(2,4)=DT
      PHIF(1,3)=DT
      PHIF(1,1)=1
      PHIF(2,2)=1
      PHIF(3,3)=1
      PHIF(4,4)=1
C           W(DM,DA)=A2/A1
      A1=Q10
      A2=Q3*Q6-Q4*Q5
      A12=A1*A2
      A1SQ=A1**2
      A1CUBE=A1**3
      A2SQ=A2**2
      PHIF(5,3)=-(2*Q3*(A12*Q6-A2SQ*2*Q3) +A2SQ*A1)*DT/A1CUBE
      PHIF(5,4)=-2*Q3*(-A12*Q5-A2SQ*2*Q4)*DT/A1CUBE
      PHIF(5,5)=1+2*Q3*Q4*A2*DT/A1SQ
      PHIF(5,6)=-2*Q3**2*A2*DT/A1SQ
      PHIF(6,3)=-2*Q4*DT*(A12*Q6-A2SQ*2*Q3)/A1CUBE
      PHIF(6,4)=-(2*Q4*(-A12*Q5-A2SQ*2*Q4)+A2SQ*A1)*DT/A1CUBE
      PHIF(6,5)=2*A2*Q4**2*DT/A1SQ
      PHIF(6,6)=1-2*A2*Q4*Q3*DT/A1SQ
      PHIF(7,7)=EXP(DELT/FTAU2)
      PHIF(8,8)=PHIF(7,7)
      DO 5010 I=1,NFS
      DO 5010 J=1,NFS
 5010 PHIF(I,J)=PHIF(J,I)
C
C        FILL IN P+ AT TIME 0
C
      PFP(1,1)=25.
      PFP(2,2)=PFP(1,1)
      PFP(3,3)=2000.
      PFP(4,4)=PFP(3,3)
      PFP(5,5)=100.
      PFP(5,5)=100.
      PFP(7,7)=.2
      PFP(8,8)=.2

C
C        FILL IN QFD AT TIME 0
C
      QFD(5,5)=SIGF1*DT
      QFD(5,5)=QFI(5,5)
      QFD (7,7) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD (8,8)=QFD (7,7)
C
C        TIME LOOP STARTS HERE
C
   50 TIME = TIME + DT
      IF(TIME.GT.TFINAL) GO TO 99
```

```
      QD(1,1)= SIGS1
      QD(2,2) = QD(1,1)
      QD(3,3) = (G1*R1)/(2.*ATAU1)
      QD(3,4) = R2*(G2/FACT2**2-.1/FACT2)-R4*C2/FACT2
      QD(3,5) = C2*R2/FACT2
      QD(4,3) = QD(3,4)
      QD(4,4) = R3*(G1/FACT3-2.*G2/FACT3**2+2.*G3/FACT3**3)-
     * R5*(G2/ATAU2+G3*DT/FACT3-2.*G3/FACT3**2)
      QD(4,5) = R3*(G3/FACT3**2-G2/FACT3)-RF*G3/FACT3
      QD(5,3) = QD(3,5)
      QD(5,4) = QD(4,5)
      QD(5,5) = R3*G3/FACT3
      DO 20 I=3,5
      DO 20 J=3,5
      QD(I+3,J+3) = QD(I,J)
      Q(I-2,J-2) = QD(I,J)
   20 CONTINUE
      WRITE(6,30)
   30 FORMAT(///2X,"THE TRUTH MODEL QD MATRIX IS:"/)
C
C        TAKING CHOLESKY SQUARE ROOT OF QD
C
      SQQD(1,1) = SQRT(QD(1,1))
      SQQD(2,2) = SQQD(1,1)
      CALL CHOLESK(Q,WORK,NIS)
      DO 33 I=1,NIS
      DO 33 J=1,NIS
      SQQD(I+2,J+2) = WORK(J,I)
      SQQD(I+5,J+5) = WORK(J,I)
   33 CONTINUE
      WRITE(6,35)
   35 FORMAT(///2X,"THE CHOLESKY SQUARE ROOT OF QD IS:"/)
      CALL MOUT(SQQD,NPS,NPS)
      IF (ISPTL.NE.1) GO TO 41
C
C        SET UP SPATIAL NOISE CORRELATION COEFFICIENT MATRIX
C
      N=64
      M=8
      DO 36 I=1,N
      R(I,I)=1.
      IF (I.GE.64) GO TO 36
      R(I,I+1)=C(1)
      IF (I.GE.63) GO TO 36
      R(I,I+2)=C(3)
      IF (I.GE.57) GO TO 36
      R(I,I+6)=C(4)
      R(I,I+7)=C(2)
      R(I,I+8)=C(1)
      R(I,I+9)=C(2)
      R(I,I+10)=C(4)
      IF (I.GE.49) GO TO 36
      R(I,I+14)=C(5)
```

```
      R(I,I+15)=C(4)
      R(I,I+16)=C(3)
      R(I,I+17)=C(4)
      R(I,I+18)=C(5)
35    CONTINUE
      DO 37 I=1,M
      R(8*I-7,8*I)=0.0
      R(8*I-7,8*I-1)=0.0
      R(8*I-6,8*I)=0.0
      IF (I.GE.8) GO TO 37
      R(8*I,8*I+1)=0.0
      R(8*I,8*I+2)=0.0
      R(8*I-1,8*I+1)=0.0
      R(8*I-7,8*I+7)=0.0
      R(8*I-7,8*I+6)=0.0
      R(8*I-6,8*I+8)=0.0
      IF (I.GE.7) GO TO 37
      R(8*I,8*I+9)=0.0
      R(8*I,8*I+10)=0.0
      R(8*I-1,8*I+9)=0.0
      IF (I.GE.6) GO TO 37
      R(8*I,8*I+17)=0.0
      R(8*I,8*I+18)=0.0
      R(8*I-1,8*I+17)=0.0
37    CONTINUE
      DO 38 I=1,N
      L=I+1
      DO 38 J=L,N
      R(J,I)=R(I,J)
38    CONTINUE
      DO 39 I=1,N
      DO 39 J=1,N
      RN(I,J)=SIGMA6*R(I,J)
39    CONTINUE
      WRITE(6,44)
44    FORMAT(///2X,"64 X 64 SPATIAL CORRELATION MATRIX:"/)
C
C        COMPUTE THE CHOLESKY SQUARE ROOT OF RN
C
      CALL CHOLESK(RN,R,64)
      WRITE(6,48)
48    FORMAT(///2X,"THE CHOLESKY SQUARE ROOT OF RN:"/)
      GO TO 42
41    DO 43 I=1,64
43    R(I,I)=1.
42    CONTINUE
      PRINT *," "
      PRINT *," "
C
      PRINT *,"          BEGIN THE MONTE CARLO SIMULATION "
C
```

```
      DO 80 L= 1,10
      TIME = 0.
      XCENTR=0.
      YCENTR=0.
      FIMIN =0.
      IMEAS=0
      FIMAX=ABS(FIMAX*0)
      AR=ARO
      SIGVF=ABS(SIGMFO)
      TRXAT=200.
      MAXIND=0
C
C        PROVIDE FILTER WITH INITIAL VELOCITY AND POSITION
C
      RHOR=(XO**2+ZO**2)
      RANGE=(RHOR+YO**2)
      XFP(3)=(ZO*XDOTO-XO*ZDOTO)/(RHOR*.00002)
      RHOR=SQRT(RHOR)
      XFP(4)=(RHOR*YDOTO-YO*((XO*XDOTO+ZO*ZDOTO)/RHOR))/(RANGE*.00002)
      PRINT *," RHOR=",RHOR," RANGE=",RANGE," XFP(3)=",XFP(3),
     *" XFP4=",XFP(4)," NRUN=",L
C
C        RESET INITIAL CONDITIONS FOR NEW RUN
C     3 FILTERS PROB(O)=.3333
      P1=.3333333
      P2=.3333333
      P3=.3333333
C        FILTER ONE INITIALLIZATION
C
      CALL INIT(XFP1,XFM1,QFD1,PFP1,PHIF1,XRES1,NFS)
      SIGF1 = 1000.
      XFP1(3)=XFP(3)
      XFP1(4)=XFP(4)
      CALL PHIFGM(PHIF1,PHIFT1,DT,DELT,FTAU2,NFS)
      CALL QFDGM(QFD1,SIGF1,DT,SIGF2,DELT,FTAU2,NFS)
      CALL PP_USO(PFP1,NFS)
C
C        FILTER TWO INITIALLIZATION
C
      CALL INIT(XFP2,XFM2,QFD2,PFP2,PHIF2,XRES2,NFS)
      SIGF1 = 300.
      XFP2(3)=XFP(3)
      XFP2(4)=XFP(4)
      CALL PHIFGM(PHIF2,PHIFT2,DT,DELT,FTAU2,NFS)
      CALL QFDGM(QFD2,SIGF1,DT,SIGF2,DELT,FTAU2,NFS)
      CALL PP_USO(PFP2,NFS)
C
C        FILTER THREE INITIALLIZATION
C
      CALL INIT(XFP3,XFM3,QFD3,PFP3,PHIF3,XRES3,NFS)
      SIGF1 = 600.
      XFP3(3)=XFP(3)
      XFP3(4)=XFP(4)
      CALL PHIFGM(PHIF3,PHIFT3,DT,DELT,FTAU2,NFS)
      CALL QFDGM(QFD3,SIGF1,DT,SIGF2,DELT,FTAU2,NFS)
```

```
      CALL PR_USC(PPP3,NPS)
      DO 46 I=1,NPS
      XS(I) = 0.
      XFM(I)=0.
   46 CONTINUE
C
C
C        TIME LOOP STARTS HERE
C
   50 TIME = TIME + DT
      PRINT *, ' TIME = ',TIME
      IF(TIME.GT.TFINAL) GO TO 99
C
C
C        PERFORM TRUTH MODEL SIMULATION
C
      VTIME=TIME-DT/2.
      IF(TIME-2.0) 500,505,505
  500 XDOT=-1000.
      YDOT=0.
      ZDOT=0.
      XVEH=X0+XDOT*TIME
      YVEH=Y0
      ZVEH=Z0
       GO TO 510
C
C
C
C        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C           THIS IS A 10-G TURN
C        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C
C
  505 XDOT=-1000.*COS(0.098*(VTIME-2.))
      YDOT=1000.*SIN(0.098*(VTIME-2.))
      XVEH=X0-2000.-10204.*SIN(0.098*(TIME-2.))
      YVEH=Y0+10204.*(1.-COS(0.098*(TIME-2.)))
      ZVEH=Z0
  510 CONTINUE
      RHOR=(XVEH**2+ZVEH**2)
      RANGE=(RHOR+YVEH**2)
      UT(1,1)=(ZVEH*XDOT-XVEH*ZDOT)/(RHOR*.00002)
      RHOR=SQRT(RHOR)
      UT(2,1)=(RHOR*YDOT-YVEH*((XVEH*XDOT+ZVEH*ZDOT)/RHOR))/(RANGE
     -*.00002)
      RANGE=SQRT(RANGE)
      VMAX=SQRT(XDOT**2+YDOT**2+ZDOT**2)/(RANGE*.00002)
      XCENTR=XCENTR+UT(1,1)*ED(1,1)
      YCENTR=YCENTR+UT(2,1)*ED(2,2)
      CALL NOISE (NPS,W)
      CALL MMPY(TEMP,SQQD,W,NPS,NPS,ONE)
      CALL MMPY(TEMP1,PHI,XS,NPS,NPS,ONE)
      CALL MADD(XS,TEMP,TEMP1,NPS,ONE,ONE)
      CALL MMPY(TEMP1,ED,UT,NPS,2,ONE)
      CALL MADD(XS,XS,TEMP1,NPS,ONE,ONE)
```

```
C
C        FILTER STATE PROPAGATION
C
      CALL MMPY(XFM1,PHIF1,XFP1,NFS,NFS,ONE)
      CALL MMPY(XFM2,PHIF2,XFP2,NFS,NFS,ONE)
      CALL MMPY(XFM3,PHIF3,XFP3,NFS,NFS,ONE)
C
C     FILTERS COVARIANCE PROPAGATION
C
      CALL PFMINUS(PFM1,PFP1,PHIF1,PHIFT1,QFC1,ONE,NFS,EXTRA1)
      CALL PFMINUS(PFM2,PFP2,PHIF2,PHIFT2,QFC2,ONE,NFS,EXTRA2)
      CALL PFMINUS(PFM3,PFP3,PHIF3,PHIFT3,QFC3,ONE,NFS,EXTRA3)
C
C        FORM CENTROID POSITION AND FILL TRUTH ARRAY
C
      IF(TIME.GT. .0-) GO TO 6
      XPEAK = XS(1)+XS(3)+XS(4)-XFM1(1)
      YPEAK = XS(2)+XS(6)+XS(7)-XFM1(2)
    6  CONTINUE
      PRINT *,"XPEAK =",XPEAK," YPEAK =",YPEAK
      CALL MEAS(XPEAK,YPEAK,4,Z,R)
C
C        SEARCH FOR FIMAX AND FIMIN
C
      IF(FIMAX0.GT.0.) GO TO 59
      FIMX0=FIMAX
      FIMAX=0.
      FIMIN=0.
      DO 55 I=1,NMS
      IF(Z(I).GT.FIMAX) FIMAX = Z(I)
   55 IF(Z(I).LT.FIMIN) FIMIN = Z(I)
      FIMAX=FIMAX+.1-*SIGVF**2-1.52*SIGVF+-.35
      FIMAX= .8*FIMX0+ .2*FIMAX
   59 CONTINUE
C
C        PERFORM MEASUREMENT UPDATE FOR THE FILTER
C        INVERSE COVARIANCE FORM
C        FORM FILTER CENTROID POSITION AND FILL OUT NON LINEAR
C        SMALL H. CALCULATE PARTIAL SMALL H PARTIAL X
C
      XPEAK1= XFM1(7)
      YPEAK1 = XFM1(8)
      XPEAK2= XFM2(7)
      YPEAK2 = XFM2(8)
      XPEAK3= XFM3(7)
      YPEAK3 = XFM3(8)
      IF((XFM(3)**2+XFM(4)**2).EQ.0.) XFM(3)=.001
      CALL MEASF(XPEAK1,YPEAK1,ONE,IF1,H1,XFM1)
      CALL MEASF(XPEAK2,YPEAK2,ONE,IF2,H2,XFM2)
      CALL MEASF(XPEAK3,YPEAK3,ONE,IF3,H3,XFM3)
      DO 60 I=1,NMS
      DO 60 J=1,NFS
      HT1(J,I)=H1(I,J)
      HT2(J,I)=H2(I,J)
      HT3(J,I)=H3(I,J)
   60 CONTINUE
```

```
C
C        CALCULATE THE RESIDUALS
C        RIH=RESIDUALS= Z-SMALL H
C
         CALL MADD(RIH1,Z,HF1,NMS,ONE,NFS)
         CALL MADD(RIH2,Z,HF2,NMS,ONE,NFS)
         CALL MADD(RIH3,Z,HF3,NMS,ONE,NFS)
C
C        COVARIANCE UPDATE P(T+)
C
         CALL PP_US(PFP1,PHIF1,HT1,-1,NFS,NMS,PFM1)
         CALL PP_US(PFP2,PHIF2,HT2,-2,NFS,NMS,PFM2)
         CALL PP_US(PFP3,PHIF3,HT3,R3,NFS,NMS,PFM3)
C
C        STATE UPDATE
C
         CALL MMPY(EXTR1,HT1,RIH1,NFS,NMS,ONE)
         CALL MMPY(EXTR2,HT2,RIH2,NFS,NMS,ONE)
         CALL MMPY(EXTR3,HT3,RIH3,NFS,NMS,ONE)
C        EXTR1= HT * (Z-SMALL H)
C
C         IF IFLAG =1,GO TO 121 FOR THAT FILTER ONLY
C        IGNORED FOR RIGHT NOW,  NEED TO RUN ONE FILTER AT A TIME
C        NOT ALL AT ONCE AS IT IS NOW SET UP
C
         IFLAG=0
         CALL XFPUP(XFP1,PFP1,EXTR1,NFS,IFLAG,RI,XFM1)
         CALL XFPUP(XFP2,PFP2,EXTR2,NFS,IFLAG,RI,XFM2)
         CALL XFPUP(XFP3,PFP3,EXTR3,NFS,IFLAG,RI,XFM3)
         CALL CONDPRE(FCOND1,PFM1,H1,HT1,NMS,NFS,RN,RIH1)
         CALL CONDPRE(FCOND2,PFM2,H2,HT2,NMS,NFS,RN,RIH2)
         CALL CONDPRE(FCOND3,PFM3,H3,HT3,NMS,NFS,RN,RIH3)
         IFLAG1=0
         IFLAG2=0
         IFLAG3=0
C
C        IFLAG = 1 IF LOWER BOUNDING OF P1, P2, P3, OR P4 OCCURS
C        IFLAG NOT BEING USED AT THIS TIME
C
         FSUM = FCOND1*P1+FCOND2*P2+FCOND3*P3
         CALL PROB(P1,FSUM,FCOND1,IFLAG1)
         CALL PROB(P2,FSUM,FCOND2,IFLAG2)
         CALL PROB(P3,FSUM,FCOND3,IFLAG3)
         PRINT *," P1= ",P1," P2= ",P2," P3= ",P3
         DO 211 I=1,NFS
         XFM(I)=XFM1(I)*P1+XFM2(I)*P2+XFM3(I)*P3
         XFP(I) = XFP1(I)*P1+XFP2(I)*P2+XFP3(I)*P3
 211     CONTINUE
         XPEAK = XS(1) + XS(3) + XS(5)  -XFM(1)
         YPEAK = XS(2) + XS(6) + XS(7)  -XFM(2)
         IF(ABS(XPEAK).GT.3.*ASPRD*SIGMS) GO TO 101
         IF(ABS(YPEAK).GT.3.*ASPRD*SIGMS) GO TO 102
C
```

```
C
C       WRITE DATA TO FILE TAPE8
C
        SAVE(1) = XS(1)
        SAVE(2) = UT(1,1)
        SAVE(3) = XS(2)
        SAVE(4) = UT(2,1)
        SAVE(5) = XFP(1)
        SAVE(6) = XFP(3)
        SAVE(7) = XFP(2)
        SAVE(8) = XFP(4)
        SAVE(9) = XCENTR
        SAVE(10) = YCENTR
        SAVE(11) = PFP(1,1)
        SAVE(12) = PFP(3,3)
        SAVE(13) = PFP(2,2)
        SAVE(14) = PFP(4,4)
        WRITE(8) SAVE
        GO TO 50
  101   PRINT *, "LOST TRACK, X CHANNEL, MEAS CALLED ",IMEAS," TIMES.
       *RUN",L," TIME=",TIME
        GO TO 105
  102   PRINT *, "LOST TRACK, Y CHANNEL, MEAS CALLED ",IMEAS," TIMES.
       *RUN",L," TIME=",TIME
  105   DO 110 J=1,14
  110   SAVE(J)=0.
  106   WRITE(8) SAVE
        TIME=TIME+DT
        IF(TIME.GT.TFINAL) GO TO 99
        GO TO 106
   99   CONTINUE
        STOP "FINISH"
        END
```

```
SUBROUTINE INIT          7.74   OPT=2                          FTN 4.8+528
```

```
        SUBROUTINE INIT(XFP,XFM,QFC,PFP,PHIF,XRES,NFS)
        DIMENSION XFP(NFS),XFM(NFS),QFC(NFS,NFS),PFP(NFS,NFS),
       *PHIF(NFS,NFS),XRES(NFS)
        DO 47 I=1,NFS
        XFP(I) = 0.
        XFM(I)=0.
        XRES(I)=0.
        DO 47 J=1,NFS
        PHIF(I,J)=0.
        QFC(I,J)=0.
        PFP(I,J) = 0.
   47   CONTINUE
        RETURN
        END
```

```
SUBROUTINE PPLUSO     74/74   OPT=2                    FTN 4.8+528

      SUBROUTINE PPLUSO(PFP,NFS)
      DIMENSION PFP(NFS,NFS)
C
C        FILL IN P+ AT TIME 0
C
      PFP(1,1)=25.
      PFP(2,2)=PFP(1,1)
      PFP(3,3)=2000.
      PFP(4,4)=PFP(3,3)
      PFP(5,5)=100.
      PFP(6,6)=100.
      PFP(7,7)=.2
      PFP(8,8)=.2
      RETURN
      END

SUBROUTINE PPMINUS    74/74   OPT=2                    FTN 4.8+528

      SUBROUTINE PPMINUS(PFM,PFP,PHIF,PHIFT,QFD,ONE,NFS,EXTRA)
      DIMENSION PFM(NFS,NFS),PFP(NFS,NFS),PHIF(NFS,NFS),PHIFT(NFS
     &QFD(NFS,NFS),EXTRA(NFS,NFS)
     *,PFPOLD(8,8),PFP(8,8)
      INTEGER ONE
C
C        FILTER COVARIANCE PROPAGATION
C
      CALL MMPY(EXTRA,PHIF,PFP,NFS,NFS,NFS)
      CALL MMPY(PFPP,EXTRA,PHIFT,NFS,NFS,NFS)
C        PFPP= PHI * P(TI-1)+ * PHIT
      CALL MADD(PFM,PFPP,QFD,NFS,NFS,ONE)
C        PFM = P(TI-)=PHI * P(TI-1)+ * PHIT + G * Q * GT
      RETURN
      END

SUBROUTINE PROB       74/74   OPT=2                    FTN 4.8+528

      SUBROUTINE PROB(P,FSUM,FCOND,IFLAG)
      P=FCOND*P/FSUM
      IF(P .GT. .01) GO TO 20
      P = .01
      IFLAG=1
   20 CONTINUE
      RETURN
      END
```

```
SU ROUTINE PPLUS      74/74   OPT=2                                  FTN 4.8+328

      SU ROUTINE PPLUS(PFP,PHIF,HT,R,NFS,NMS,PFM)
      DIMENSION PFP(NFS,NFS),PHIF(NFS,NFS),H(NMS,NFS),HT(NFS,NMS)
     *WKAREA(50,50),EXTRA(8,8)
     *,PFPOLD(8,8)
     *,PFM(8,8)
      CALL MMPY(PFP,HT,H,NFS,NMS,NFS)
C        PFP= HT * H  (R-1 IS SCALAR AND MULTIPLIED LATER)
      NFS2 = NFS*NFS
      CALL SHIFTA(PFP,PFPOLD,NFS2,NFS2)
      RI=.5
      IDGT = 3
      CALL LINV2F(PFM,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
C        EXTRA= P(TI-)-1
      DO 65 I=1,NFS
      DO 65 J=1,NFS
      PFP(I,J)=PFP(I,J)*RI + EXTRA(I,J)
   65 CONTINUE
C        PFP= P(TI+)-1= HT * R-1 * H + P(TI-)-1
      IDGT = 3
      CALL LINV2F(PFP,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
      DO 70 I=1,NFS
      DO 70 J=1,NFS
      PFP(I,J) = EXTRA(I,J)
   70 CONTINUE
      DO 130 I=1,NFS
      IF(PFP(I,I).GT.0.) GO TO 130
      PRINT *,"PFP(",I,I,")=",PFP(I,I)
      PFP(I,I)=PFPOLD(I,I)
  130 CONTINUE
C        PFP=P(TI+)
      RETURN
      END

SU ROUTINE PHIFGM     74/74   OPT=2                                  FTN 4.8+328

      SUBROUTINE PHIFGM(PHIF,PHIFT,DT,DELT,ETAU2,NFS)
      DIMENSION PHIF(NFS,NFS),PHIFT(NFS,NFS)
      PHIF(1,1)=1.
      PHIF(1,3)=DT
      PHIF(1,5)=DT**2/2.
      PHIF(2,2)=PHIF(1,1)
      PHIF(2,4)=DT
      PHIF(2,5)=PHIF(1,5)
      PHIF(3,3)=1.
      PHIF(3,5)=DT
      PHIF(4,4)=1.
      PHIF(4,5)=DT
      PHIF(5,5)=1.
      PHIF(6,6)=1.
      PHIF(7,7)=EXP(DELT/ETAU2)
      PHIF(8,8)=PHIF(7,7)
      DO 56 I=1,NFS
      DO 56 J=1,NFS
   56 PHIFT(I,J)=PHIF(J,I)
      RETURN
      END
```

```
      SUBROUTINE QFD(NFS,QFD,SIGF1,DT,SIGF2,DELT,ETAU2,NFS)
      DIMENSION QFD(NFS,NFS)
C
C        FILL OUT FILTER DISCRETE QFD MATRIX
C        FOR START OF ACQUISITION PHASE
C
      QFD(1,1)=DT**5*SIGF1/20.
      QFD(1,3)=DT**4*SIGF1/8.
      QFD(1,5)=DT**3*SIGF1/6.
      QFD(2,2)=QFD(1,1)
      QFD(2,4)=QFD(1,3)
      QFD(2,6)=QFD(1,5)
      QFD(3,1)=QFD(1,3)
      QFD(3,3)=DT**3*SIGF1/3.
      QFD(3,5)=DT**2*SIGF1/2.
      QFD(4,2)=QFD(3,1)
      QFD(4,4)=QFD(3,3)
      QFD(4,6)=QFD(3,5)
      QFD(5,1)=QFD(1,5)
      QFD(5,3)=QFD(3,5)
      QFD(5,5)=SIGF1*DT
      QFD(6,2)=QFD(1,5)
      QFD(6,4)=QFD(3,5)
      QFD(6,6)=QFD(5,5)
      QFD(7,7)=(SIGF2**2)*(1.-EXP(2.*DELT/ETAU2))
      QFD(8,8)=QFD(7,7)
      RETURN
      END
```

SUBROUTINE XFPUP 74/74 OPT=2 FTN 4.8+528

```
      SUBROUTINE XFPUP(XFP,PFP,EXTRA,NFS,IFLAG,RI,XFM)
      DIMENSION XFP(NFS),PFP(NFS,NFS),EXTRA(NFS),
     *DX(8),DXDXT(8,8),XFM(8)
      INTEGER ONE
      ONE = 1
      CALL MMPY(DX,PFP,EXTRA,NFS,NFS,ONE)
C        DX = P(TI+) * HT * (Z - SMALL H)
      DO 75 I=1,NFS
      DX (I)=DX (I)*RI
   75 CONTINUE
C        DX =DELTA X = P(TI+) * HT * R-1 * (Z - SMALL H)
      DO 78 I=1,NFS
      DO 78 J=1,NFS
      DXDXT(I,J)=DX (I)*DX (J)
   78 CONTINUE
C        DXDXT= (X(TI+) - X(TI-)) * (X(TI+) - X(TI-))T
      TRXXTO=200.
      TRXXT=0.
      DO 79 I=1,NFS
   79 TRXXT=TRXXT+DXDXT(I,I)
      MAXIND=1
      TRXXT=.8*TRXXTO+.2*TRXXT
      CALL MADD(XFP,DX,XFM,NFS,ONE,ONE)
C        XFP=X(TI+)= P(TI+) * R-1 * HT * (Z - SMALL H) + X(TI-)
  222 RETURN
      END
```

```
SUBROUTINE CONPRE    7/77.    OPT=2                    FTN 4.6+532

      SUBROUTINE CONPRE(FCOND,PFM,HT,NMS,NFS,RN,XRES)
      INTEGER ONE
      DIMENSION PFM(NFS,NFS),H(NMS,NFS),HT(NFS,NMS),XRES(NMS),
     *XREST(1,64),AJ(64,64),HPFM(64,8),A(64,64),
     *HPFMHT(64,64),AINV(64,64),RTA(1,64),RTAR(1,1)
     *,AINVERS(64,64)
C
C       AJ=H*P(T-)*HT+R
C
      ONE=1
      CALL MMPY(HPFM,H,PFM,NMS,NFS,NFS)
      CALL MMPY(HPFMHT,HPFM,HT,NMS,NFS,NMS)
      CALL MADD(AJ,HPFMHT,RN,NMS,NMS,ONE)
      DO 5 I=1,NMS
      DO 5 J=1,NMS
      A(I,J)=0.0
      AINVERS(I,J)=0.0
    5 CONTINUE
C
C       TAKE DIAGONAL OF AJ MATRIX AS REPRESENTATIVE OF THE WHOLE
C       MATRIX TO AVOID A 64 X 64 MATRIX INVERSION
      DO 6 I=1,NMS
      A(I,I)=AJ(I,I)
    6 CONTINUE
C
C       THE DETERMINANT OF "A" IS "DETA"
C
      DETA=1.
C       AINVERS = A**(-1)
      DO 9 I=1,NMS
      AINVERS(I,I)=(DETA/A(I,I))/DETA
    9 CONTINUE
      DO 10 J=1,NMS
       XREST(1,J)=XRES(J)
   10  CONTINUE
C
C       F(T)=1/((2*3.14159**M/2)*ABS(AJ)**.5*EXP(-)
C       (-)=(-.5*XREST*(AJ**-1)*XRES) ; M=# OF FILTERS
C
      CALL MMPY(RTA,XREST,AINVERS,ONE,NMS,NMS)
      CALL MMPY(RTAR,RTA,XRES,ONE,NMS,ONE)
      RAR=RTAR(1,1)
      RAR = .01*RAR
      IF(RAR .LT. -460.) RAR=-460.
      IF(RAR .GT. +460.) RAR = 460.
      RAR=-.5*RAR
      EXPO=EXP(RAR)
      FCOND = EXPO
      RETURN
      END
```

```
SUBROUTINE MEAS      74/74   OPT=1                              FTN 4.8+528

      SUBROUTINE MEAS (XPEAK,YPEAK,ISUB,Z,R)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
     *,ASPRO,FIMAX,UT(2,1),IMEAS,AR,SIGVF,PL2P(64,2),VMAX,SIGMFO
     *,RANGEO,RANGE,SIGPVF,CSTH,SNTH
      REAL IMAX
      DIMENSION Z(8,8), R(64,64) ,WI(64,1),WD(64,1)
      ZMIN = 0.
      SIGPV=SIGMS*RANGEO/RANGE
      PLVEL=SQRT(UT(1,1)**2+UT(2,1)**2)
      SNTH=UT(2,1)/PLVEL
      CSTH=UT(1,1)/PLVEL
      SIGV=(1.+(ASPRO-1.)*PLVEL/VMAX)*SIGPV
      I = (NPIX*ISUB)
      IDIV = ISUB**2
      XINCR = XFOV/FLOAT(I)
      YINCR = YFOV/FLOAT(I)
      X = -1.*XFOV/2. + XINCR/2.
      Y = YFOV/2. - YINCR/2.
      XO = X
      CALL NOISE(64,W1)
      CALL MMPY(WD,R,W1,64,64,1)
      MN=0
      DO 20 K=1,NPIX
      DO 15 J=1,NPIX
      MN=MN+1
      TOTAL = 0.
      XN = X
      YN = Y
      DO 10 N=1,ISUB
      YCSTH=(YN-YPEAK)*CSTH
      YSNTH=(YN-YPEAK)*SNTH
      DO 5 M =1,ISUB
      ARGSPV=YCSTH-(XN-XPEAK)*SNTH
      ARGSV=(XN-XPEAK)*CSTH+YSNTH
      ARG=-((ARGSV/SIGV)**2+(ARGSPV/SIGPV)**2)*.5
      TOTAL =TOTAL+EXP(ARG)*IMAX
      XN = X +FLOAT(M)*XINCR
    5 CONTINUE
      YN = Y- FLOAT(N)*YINCR
      XN = X
   10 CONTINUE
      Z(K,J)=TOTAL/FLOAT(IDIV)
C
C        ADD BACKGROUND AND FLIR NOISE BOTH ZERO MEAN
C
      IF(SIGFLR.EQ.0.) GO TO 30
      GAUSS=0.
      DO 110 LL=1,12
      GAUSS=GAUSS+RANF(DUM1)
  110 CONTINUE
      F=(GAUSS-6.)*SIGFLR
      Z(K,J) = Z(K,J) + F
   30 Z(K,J)=Z(K,J)+WD(MN,1)
```

```
                IF(Z(K,J).LT.ZMIN)ZMIN=Z(K,J)
55              X = XN +FLOAT(ISUB)*XINCR
             15 CONTINUE
                Y = Y - FLOAT(ISUB)*YINCR

                X=X0
             20 CONTINUE
60              IMEAS=IMEAS+1
                IF(ZMIN.EQ.0.)RETURN
                DO 25 I=1,NPIX
                DO 25 J=1,NPIX
                Z(I,J) = Z(I,J)-ZMIN+.1
65           25 CONTINUE
                RETURN
                END
```

SUBROUTINE SHIFT 74/74 OPT=1 FTN 4.8+528

```
 1              SUBROUTINE SHIFT(A,B,IS,N)
                DIMENSION A(N),B(N)
          C       A=INPUT ARRAY,B=OUTPUT ARRAYN=ARRAY SIZE,IS=AMT OF SHIF
          C
 5              NN=N
                DO 300 I=1,N
            300 B(I)=0.
          C       TEST FOR LEFT OR RIGHT SHIFT
                IF(IS.LE.0) GO TO 100
10        C       EXECUTE RIGHT SHIFT
                MM=1+IS
                M=N-IS
                CALL SHIFTA(A(1),B(MM),M,NN)
                GO TO 200
15        C       EXECUTE LEFT SHIFT
            100 MM=1-IS
                M=N+IS
                CALL SHIFTA(A(MM),B(1),M,NN)
            200 RETURN
20              END
```

SUBROUTINE SHIFTA 74/74 OPT=1 FTN 4.8+528

```
 1              SUBROUTINE SHIFTA(A,B,M,N)
                DIMENSION A(N),B(N)
                DO 100 K=1,M
            100 B(K) = A(K)
 5              RETURN
                END
```

```
SUBROUTINE MEASF      74/74   OPT=1                          FTN 4.8+528

1                     SUBROUTINE MEASF(XPEAK,YPEAK,ISUB,Z,H,XFM,
                      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMA3,SIGFLR,
                     *,ASPRO,FIMAX,UT(2,1),IMEAS,AR,SIGVF,PL2P(64,2),VMAX,SIGMF0
                     *,RANGE0,RANGE,SIGPVF,CSTH,SNTH
5                      DIMENSION Z(8,8),H(64,8),XFM(8)
                      REAL IMAX
                      ZMIN = 0.
                      PLVEL= SQRT(XFM(3)**2+XFM(4)**2)
                      SNTH= XFM(4)/PLVEL
10                    CSTH= XFM(3)/PLVEL
                      SIGPVF=SIGVF/AR
                      I = (NPIX*ISUB)
                      IDIV = ISUB**2
                      XINCR = XFOV/FLOAT(I)
15                    YINCR = YFOV/FLOAT(I)
                      X = -1.*XFOV/2. + XINCR/2.
                      Y = YFOV/2. - YINCR/2.
                      XO = X
                      DO 20 K=1,NPIX
20                    NUM = K
                      DO 15 J=1,NPIX
                      TOTAL = 0.
                      SUM1=0.
                      SUM2=0.
25                    XN = X
                      YN = Y
                      DO 10 N=1,ISUB
                      YCSTH=(YN-YPEAK)*CSTH
                      YSNTH=(YN-YPEAK)*SNTH
30                    DO 5 M =1,ISUB
                      ARGSPV=YCSTH-(XN-XPEAK)*SNTH
                      ARGSV=(XN-XPEAK)*CSTH+YSNTH
                      ARG=-((ARGSV/SIGVF)**2+(ARGSPV/SIGPVF)**2)*.5
                      PART= EXP(ARG)*FIMAX
35                    TOTAL = TOTAL+PART
                      SUM1 = SUM1 + PART*(ARGSV*CSTH/SIGVF**2-ARGSPV*SNTH/SIGPVF**
                      SUM2 = SUM2 + PART*(ARGSPV*CSTH/SIGPVF**2+ARGSV*SNTH/SIGVF**
                      XN = X +FLOAT(M)*XINCR
                    5 CONTINUE
40                    YN = Y-FLOAT(N) * YINCR
                      XN = X
                   10 CONTINUE
                      Z(K,J) = TOTAL/FLOAT(IDIV)
                      IF(SIGMF0.GT.0.) GO TO 16
45                    PL2P(K+(J-1)*8,1)= PART*(-(ARGSPV/SIGVF)**2*AR)
                      PL2P(K+(J-1)*8,2)= PART*((ARGSV**2+ARGSPV**2*AR**2)/SIGVF**
                   16 CONTINUE
                      IF(Z(K,J).LT.ZMIN)ZMIN=Z(K,J)
                      H(NUM,1) = SUM1/FLOAT(IDIV)
55                    H(NUM,2) = SUM2/FLOAT(IDIV)
                      H(NUM,3)=0.
                      H(NUM,4)=0.
                      H(NUM,5)=0.
```

```
            H(NUM,6)=0.
            H(NUM,7)=H(NUM,1)
            H(NUM,8)=H(NUM,2)
            X = XN +FLOAT(ISUB)*XINCR
            NUM = NUM + NPIX
         15 CONTINUE
            Y = Y - FLOAT(ISUB)*YINCR
            X=X0
         20 CONTINUE
            IF(ZMIN.EQ.0.)RETURN
            DO 25 I=1,NPIX
            DO 25 J=1,NPIX
            Z(I,J) = Z(I,J)-ZMIN+.1
         25 CONTINUE
            RETURN
            END
```

SUBROUTINE MADD 74/74 OPT=1 FTN 4.8+528

```
      SUBROUTINE MADD(C,A,B,J,K,IFLAG)
      DIMENSION A(J,K),B(J,K),C(J,K)
      IF(IFLAG.EQ.1) GO TO 6
      DO 5 N=1,J
      DO 5 M=1,K
      C(N,M) = A(N,M) - B(N,M)
    5 CONTINUE
      RETURN
    6 DO 10 N=1,J
      DO 10 M=1,K
      C(N,M) = A(N,M) + B(N,M)
   10 CONTINUE
      RETURN
      END
```

FUNCTION DOT 74/74 OPT=1 FTN 4.8+528

```
      FUNCTION DOT(NR,A,B)
      DIMENSION A(1),B(1)
      DOT=0.
      DO 1 I=1,NR
    1 DOT = DOT+A(I)*B(I)
      RETURN
      END
```

```
  SUBROUTINE LOCATE     74/74   OPT=1                          FTN 4.8+528

 1                SUBROUTINE LOCATE(R,RD,REF,XR,XTR,T,TD,C,E,KK,KD)
                  DIMENSION R(KK),RD(KD),T(KK),C(3),TD(KD),REF(KK)
               CALCULATE THE TARGET CENTROID POSITION
                  KT=KK
 5                KS=KD
                  CALL CENTR(T,XT,KT)
               CALCULATE A ROUGH ESTIMATE OF TARGET POSITION OFFSET
                  XTR=XT-XR
               C  ROUND OFF ANSWER TO NEAREST INTEGER
10                CALL RNDF(XTR,MARK)
               COMPUTE TARGET DENOMINATOR
                  CALL RDEN(T,TD,KT,KS)
                  DO 100 JJ=1,3
                100 C(JJ)=0.
15                J=0
                  IS=MARK-2
                  N=KK
               C   START MAIN LOOP HERE          **********************
                  200 CONTINUE
20                J=J+1
                  IS=IS+1
                  IF(IS.GE.8.OR.IS.LE.-8) GO TO 525
               C   SHIFT REFERENCE DATA
                  CALL SHIFT(R,REF,IS,N)
25             CALCULATE THE SQUARE OF THE CORRELATION COEFFICIENTS
                  SUM=0.
                  DO 300 JJ=1,N
                  SUM=SUM+T(JJ)*REF(JJ)
                300 CONTINUE
30                CA=SUM**2
                  DENOM=TD(N-IS)*RD(N+IS)
                  IF (DENOM.LT.1.E-8)PRINT*,"N=",N,"IS=",IS,"TD=",TD(N-IS),
                 *"RD=",RD(N+IS),"SUM=",SUM,"MARK=",MARK
                  C(J)=CA/DENOM
35                IF(C(J).GT.1.) C(J) = 1.
               C   TEST TO SEE IF C(2) HAS BEEN COMPUTED
                  IF(C(2).LE.0.)GO TO 200
                  IF(C(2).GE.C(1)) GO TO 400
               C   MISALIGNMENT, SHIFT LEFT)
40                C(3)=C(2)
                  C(2)=C(1)
                  J=0
                  MARK=MARK-1
                  IS=IS-3
45                GO TO 200
                400 CONTINUE
                  IF(C(3).LE.0.) GO TO 200
                  IF(C(3).LE.C(2)) GO TO 500
               C   MISALIGNMENT SHIFT RIGHT
50                C(1)=C(2)
                  C(2)=C(3)
                  MARK=MARK+1
                  J=2
```

```
                GO TO 200
  500 CONTINUE
C   END OF MAIN LOOP SECTION
C
COMPUTE TARGET-REFERENCE PRECISION OFFSET ESTIMATE
C
      DX=0.
      IF(C(1).EQ.0.0.OR.C(2).EQ.0.0.OR.C(3).EQ.0.) GOTO 550
      DN=2.*(1./C(1) +1./C(3) -2./C(2))
      IF(DN.GT.0.) GO TO 600
  525 E=1000.
      GO TO 700
  550 PRINT*,"WARNING ********** CORRELATION  COEFF = 0. "
      DN=2.*(2.*C(2)-C(1)-C(3))
      DX = (C(3)-C(1))/DN
      E = FLOAT(MARK)+DX
      GO TO 700
  600 CONTINUE
      DX=(1./C(1) -1./C(3))/DN
      E=FLOAT(MARK)+DX
  700 RETURN
      END
```

SUBROUTINE NOISE 74/74 OPT=1 FTN 4.8+528

```
      SUBROUTINE NOISE(N,W)
      DIMENSION W(N)
      DO 15 J=1,N
      TOTAL=0.
      DO 5 I=1,12
      TOTAL =TOTAL + RANF(DUM)
    5 CONTINUE
      W(J) = TOTAL - 6.
   15 CONTINUE
      RETURN
      END
```

SUBROUTINE MOUT 74/74 OPT=1 FTN 4.8+528

```
      SUBROUTINE MOUT(A,NR,NC)
      DIMENSION A(NR,NC)
      DO 10 I=1,NR
      WRITE(6,5) (A(I,J),J=1,NC)
    5 FORMAT(2X,8(G13.7,3X))
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CORR(RA,TA,I1,J1,XCORR,YCORR,CX,CY)
      DIMENSION RA(I1,J1),TA(I1,J1),R(8),T(8),TARG(8),C(3),RD(15),
     * TD(15),REF(8)
C
C        PERFORM ROW CORRELATION
C
      CX=.95
      CY=.95
    1 COUNT=0.
      SUMX=0.
      N=J1
      K=2*N-1
      EE=1.E-8
      DO 1200 II=1,I1
      DO 1100 JJ=1,J1
      R(JJ) = RA(II,JJ)+EE
      T(JJ) = TA(II,JJ)+EE
 1100 CONTINUE
      CALL RDEN(R,RD,N,K)
      CALL CENTR(R,XR,N)
      CALL LOCATE(R,RD,REF,XR,XTR,T,TD,C,E,N,K)
      IF(C(2).GT.CX) GO TO 2100
      TARG(II) = 1000.
      GO TO 2200
 2100 CONTINUE
      TARG(II) = C
 2200 CONTINUE
 1200 CONTINUE
      DO 3000 I=1,J1
      IF(TARG(I) .EQ.1000.) GO TO 3000
      SUMX=SUMX+TARG(I)
      COUNT=COUNT+1.
 3000 CONTINUE
      IF(COUNT.EQ.0.0) GO TO 3500
      XCORR=SUMX/COUNT
C
C        PERFORM CORRELATION ON COLUMNS
C
    2 COUNT=0.
      SUMY=0.
      N=I1
      K=2*N-1
      DO 4200 JJ=1,J1
      DO 4100 II=1,I1
      R(II) = RA(II,JJ) +EE
      T(II) = TA(II,JJ) +EE
 4100 CONTINUE
      CALL RDEN(R,RD,N,K)
      CALL CENTR(R,XR,N)
      CALL LOCATE(R,RD,REF,XR,XTR,T,TD,C,E,N,K)
      IF(C(2).GT.CY) GO TO 3100
      TARG(JJ) = 1000.
      GO TO 3200
```

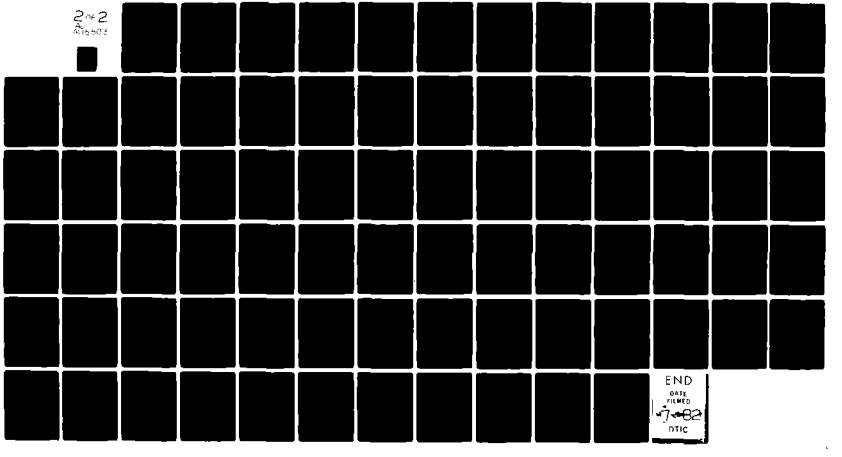
```
 3100 CONTINUE
      TARG(JJ) =E
 3200 CONTINUE
 4200 CONTINUE

      DO 4000 I=1,I1
      IF(TARG(I).EQ.1000.) GO TO 4000
      SUMY =SUMY + TARG(I)
      COUNT=COUNT+1.
 4000 CONTINUE
      IF(COUNT.EQ.0.0) GO TO 5000
      YCORR = SUMY/COUNT
      YCORR = -YCORR
      RETURN
 5000 CONTINUE
      CX=CX-.025
      IF(CX.LT.0.5) PRINT*,"********** CX<0.5 **********"
      GO TO 1
 5000 CONTINUE
      CY=CY-.025
      IF(CY.LT.0.5) PRINT*,"********** CY<0.5 **********"
      GO TO 2
      END
```

```
SUBROUTINE RDEN        74/74   OPT=1                    FTN 4.8+528

 1            SUBROUTINE RDEN(R,D,N,K)
              C     THIS SUBROUTINE COMPUTES THE CORRELATION DENOMINATORS
              C     FOR THE REFERENCE FUNCTION.
              C        R= INPUT REFERENCE ARRAY
 5            C        D = OUTPUT DENOMINATOR ARRAY
              C        N IS THE INPUT ARRAY DIMENSION AND ALSO THE OUTPUT ARRAY
              C        MARKER POSITION.
              C        K IS THE OUTPUT ARRAY DIMENSION
                   DIMENSION R(N),D(K)
10                 SUM = 0.
                   J=1
                   JJ=K
               200 SUM = SUM+R(J)**2
                   D(JJ) = SUM
15                 IF (JJ.EQ.J )GO TO 300
                   JJ=JJ-1
                   J=J+1
                   GO TO 200
               300 J=K
20             400 JJ=JJ-1
                   D(JJ) = SUM-D(J)
                   J=J-1
                   IF(JJ.NE.1) GO TO 400
                   RETURN
25                 END
```

AD-A115 503 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 17/8
ALTERNATIVE DYNAMICS MODELS AND MULTIPLE MODEL FILTERING FOR A --ETC(U)
DEC 81 P M FLYNN
UNCLASSIFIED AFIT/GE/EE/81D-21 NL



```
SUBROUTINE CHOLESK     74/74   OPT=1                          FTN 4.8+528

 1                    SUBROUTINE CHOLESK(A,S,N)
                      DIMENSION A(1),S(1)
                      NDIM=N
                      NDIM1=N+1
 5                    TOL=1.E-6
                      MR=0
                      NN=N*NDIM
                      TOL1=0.
                      DO 1 I=1,NN,NDIM1
 10                   R=ABS(A(I))
                1     IF(R.GT.TOL1)TOL1=R
                      TOL1 = TOL1*1.E-12
                      II=1
                      DO 50 I=1,N
 15                   IM1 = I-1
                      DO 5 JJ=I,NN,NDIM
                5     S(JJ) = 0.
                      ID = II+IM1
                      R=A(ID)-DOT(IM1,S(II),S(II))
 20                   IF(ABS(R).LT.(TOL*A(ID)+TOL1)) GO TO 50
                      IF(R) 15,50,20
                15    MR=-1
                      WRITE(6,1000)
               1000   FORMAT(37H TRIED TO FACTOR AN INDEFINATE MATRIX    )
 25                   RETURN
                20    S(ID) = SQRT(R)
                      MR=MR+1
                      IF(I.EQ.N) RETURN
                      L=II+NDIM
 30                   DO 25 JJ=L,NN,NDIM
                      IJ=JJ+IM1
                25    S(IJ) = (A(IJ)-DOT(IM1,S(II),S(JJ)))/S(ID)
                50    II=II+NDIM
                      RETURN
 35                   END
```

```
SUBROUTINE MMPY        74/74   OPT=1                          FTN 4.8+528

 1                    SUBROUTINE MMPY(C,A,B,K,M,N)
                      DIMENSION C(K,N),A(K,M),B(M,N)
                      DO 1 I=1,K
                      DO 1 J=1,N
 5                    C(I,J)=0.
                1     CONTINUE
                      DO 5 L=1,K
                      DO 5 J=1,N
                      DO 5 I=1,M
 10                   C(L,J) = C(L,J) + (A(L,I)*B(I,J))
                5     CONTINUE
                      RETURN
                      END
```

SUBROUTINE CENTR 74/74 OPT=1 FTN 4.8+528

```
1                SUBROUTINE CENTR(A,X,J)
               C     THIS SUBROUTINE COMPUTES THE CENTROID OF THE ARRAY A
               C     THE OUTPUT VARIABLE IS:X, AND THE ARRAY DIMENSION IS:J
                     DIMENSION A(J)
5                    SUM = 0.
                     S = 0.
                     X = 1.
                     DO 100 I=1,J
                     SUM = SUM + FLOAT(I)*A(I)
10             100   S=S+A(I)
                     IF(S.LE.0.) GO TO 200
                     X=SUM/S
               200   RETURN
                     END
```

SUBROUTINE RNDF 74/74 OPT=1 FTN 4.8+528

```
1                SUBROUTINE RNDF(A,K)
               C     SUBROUTINE TO ROUNDOFF A,TO NEAREST INTEGER,K
               C
                     K=IFIX(A)
5                    B=A-K
                     IF(B.LT.0.5)GO TO 100
                     K=K+1
               100   RETURN
                     END
```

SUBROUTINE MWRITE 74/74 OPT=1 FTN 4.8+528

```
1                SUBROUTINE MWRITE (A,N,M,IDIM)
                 DIMENSION A(IDIM,1)
                 PRINT *," "
                 DO 350 I4=1,N
5                WRITE (6,340) (A(I4,J4),J4=1,M)
               340 FORMAT (1X,16(F7.4,1X))
               350 CONTINUE
                 PRINT *," "
                 END
```

APPENDIX C

Performance Plots for the Drownian Motion Filter

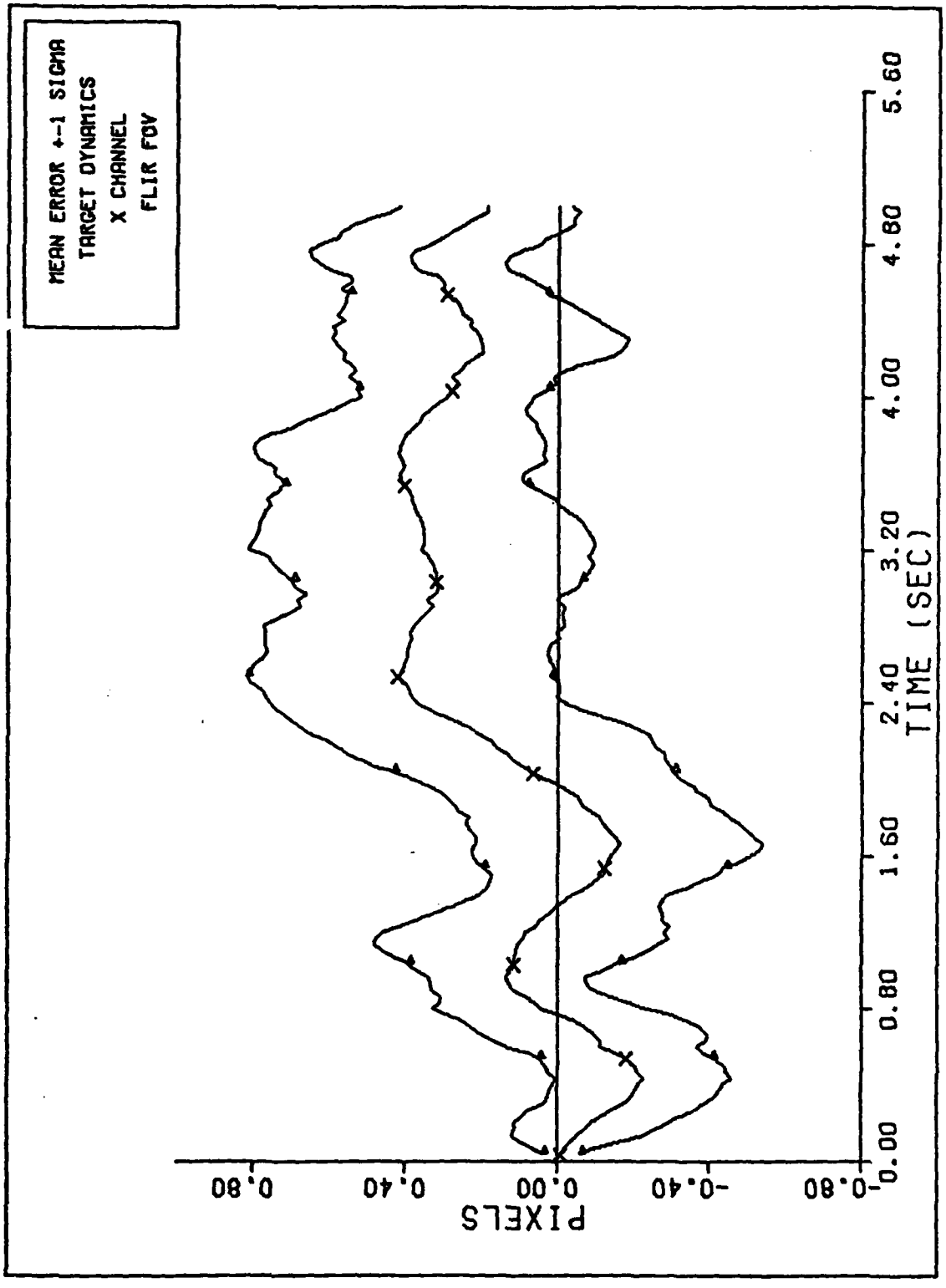
This appendix contains the plotted outputs of the performance of the BM filter. Three trajectories were simulated and eight plots are included for each case. They are:

--mean error $\pm$ 1 sigma of the filter estimate of the 'x' and 'y' target position. These plots were used for checking the mean bias error and rms position error in general.

--mean error $\pm$ 1 sigma of the filter estimate of the 'x' and 'y' target velocity. These plots were used to check any mean bias errors and rms velocity errors in general.

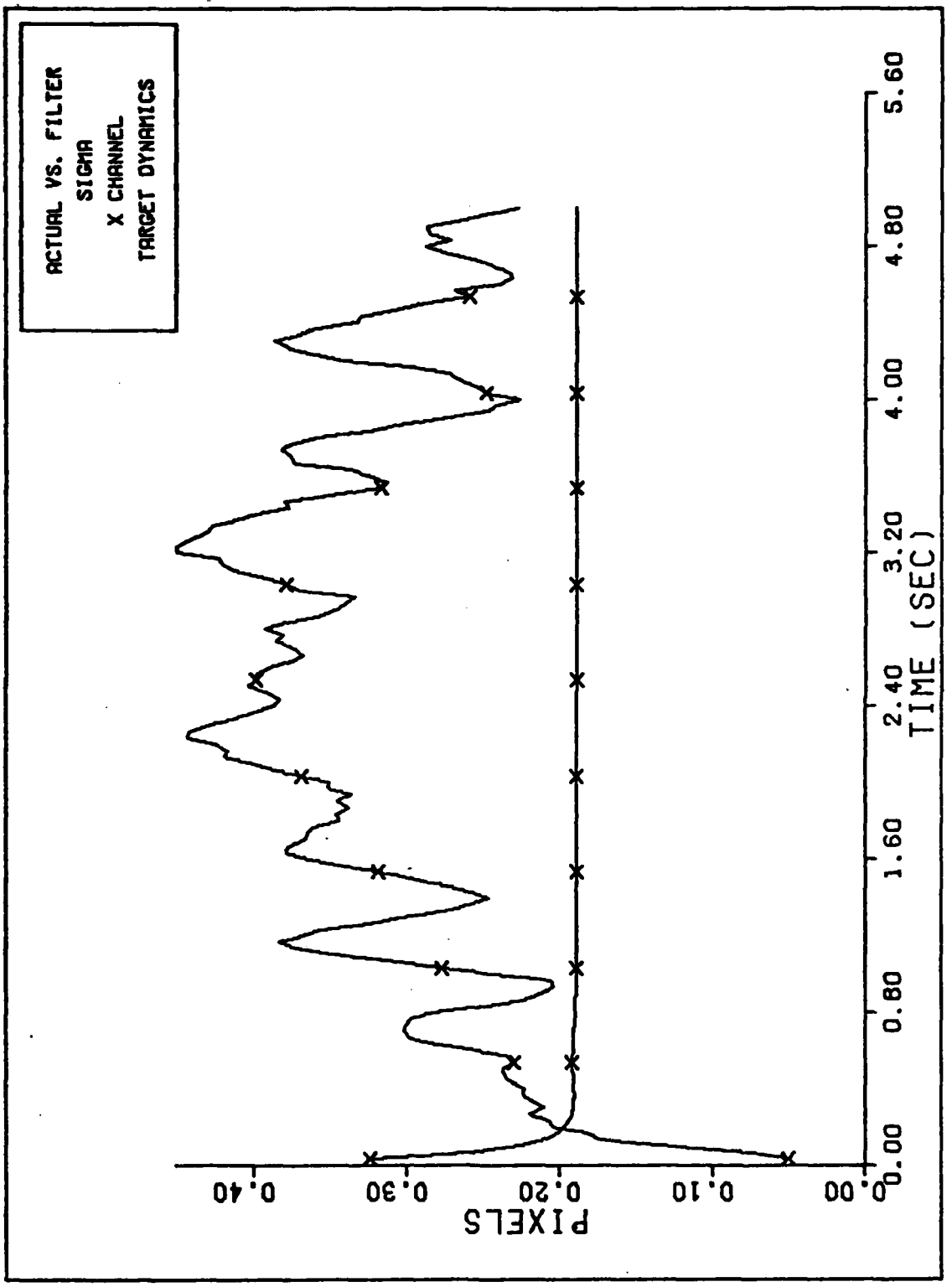
--true and filter-indicated standard deviation of the 'x' and 'y' position. These plots were used in tuning the filter to the various trajectories simulated.

--true and filter-indicated standard deviation of the 'x' and 'y' target velocity. These plots were used in tuning the filter to the various trajectories simulated.



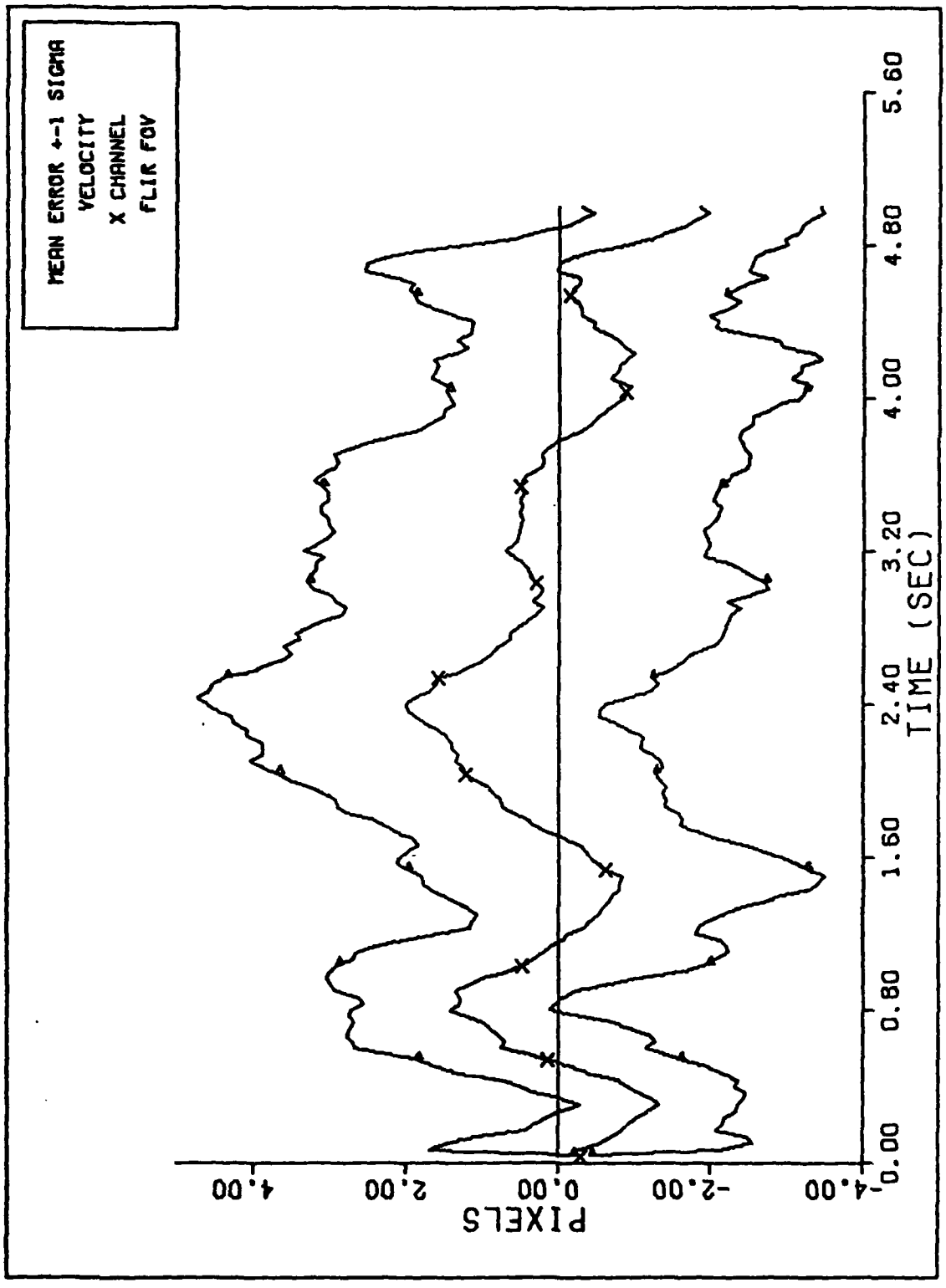
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure C-1 20 g Q=600 Performance Plot



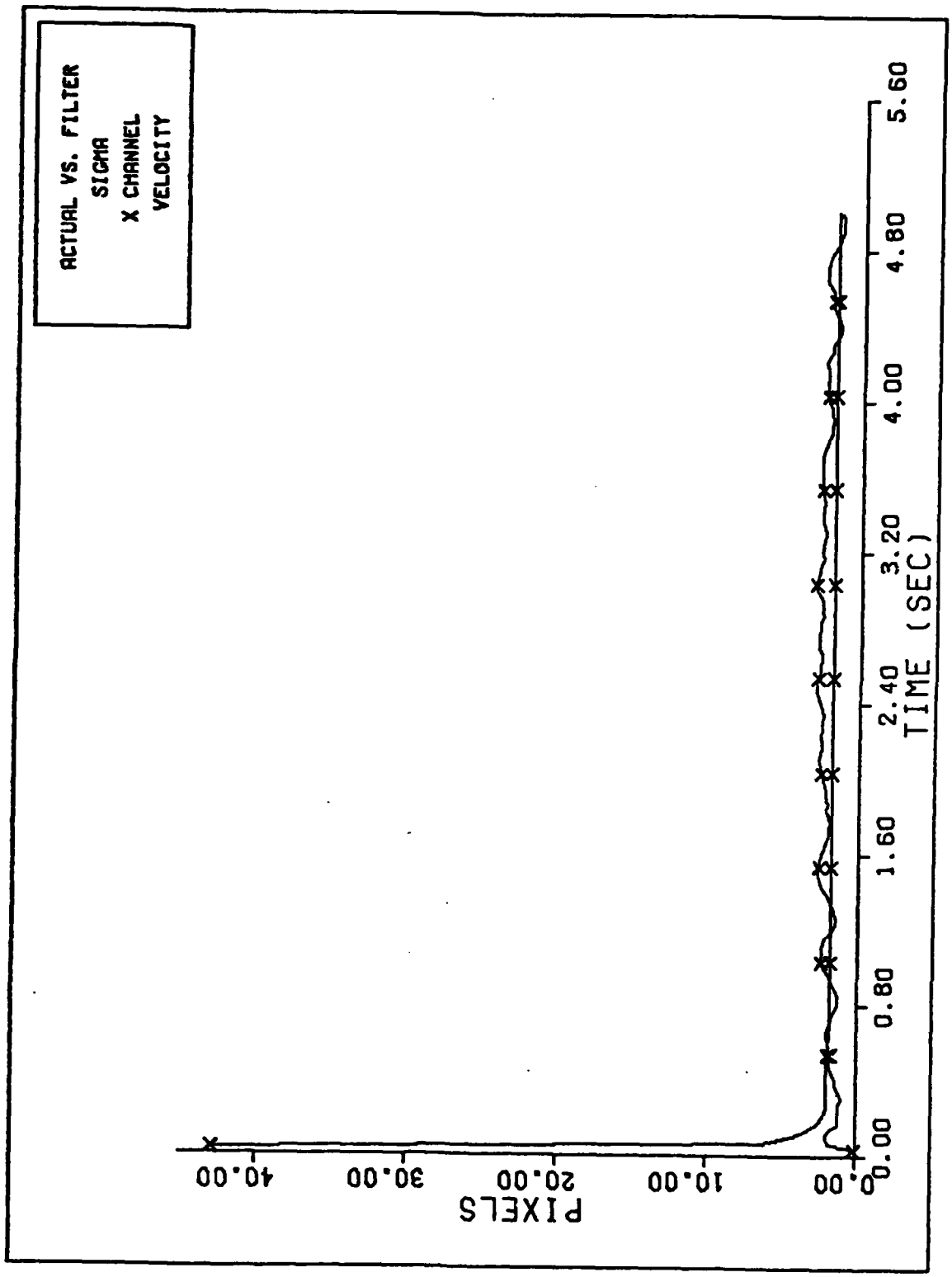
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-2 20 g Q=600 Performance Plot



X CHANNEL VELOCITY ERROR (S/N= 12.9

Figure C-3 20 g Q=600 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-4 20 g Q=600 Performance Plot

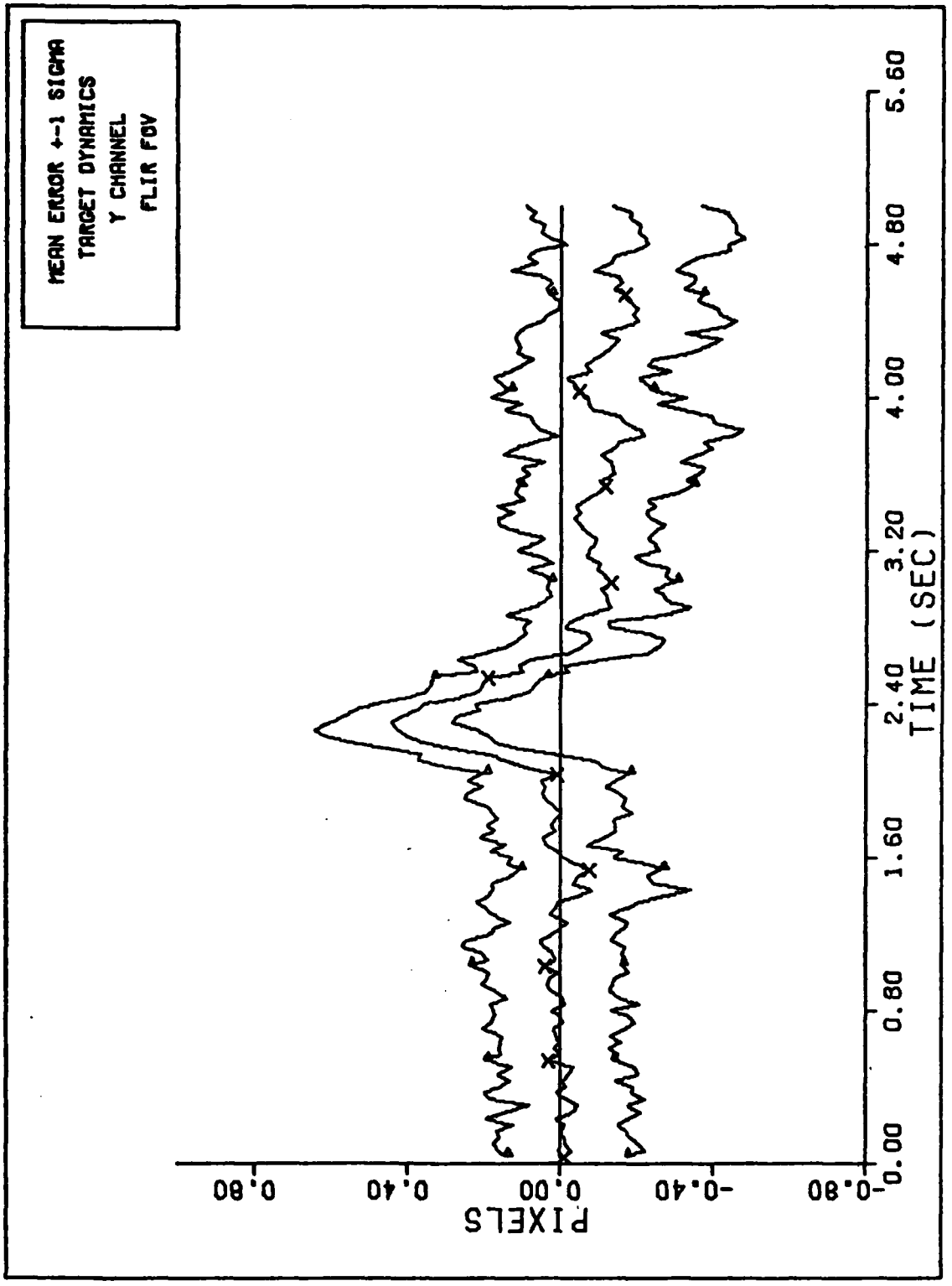
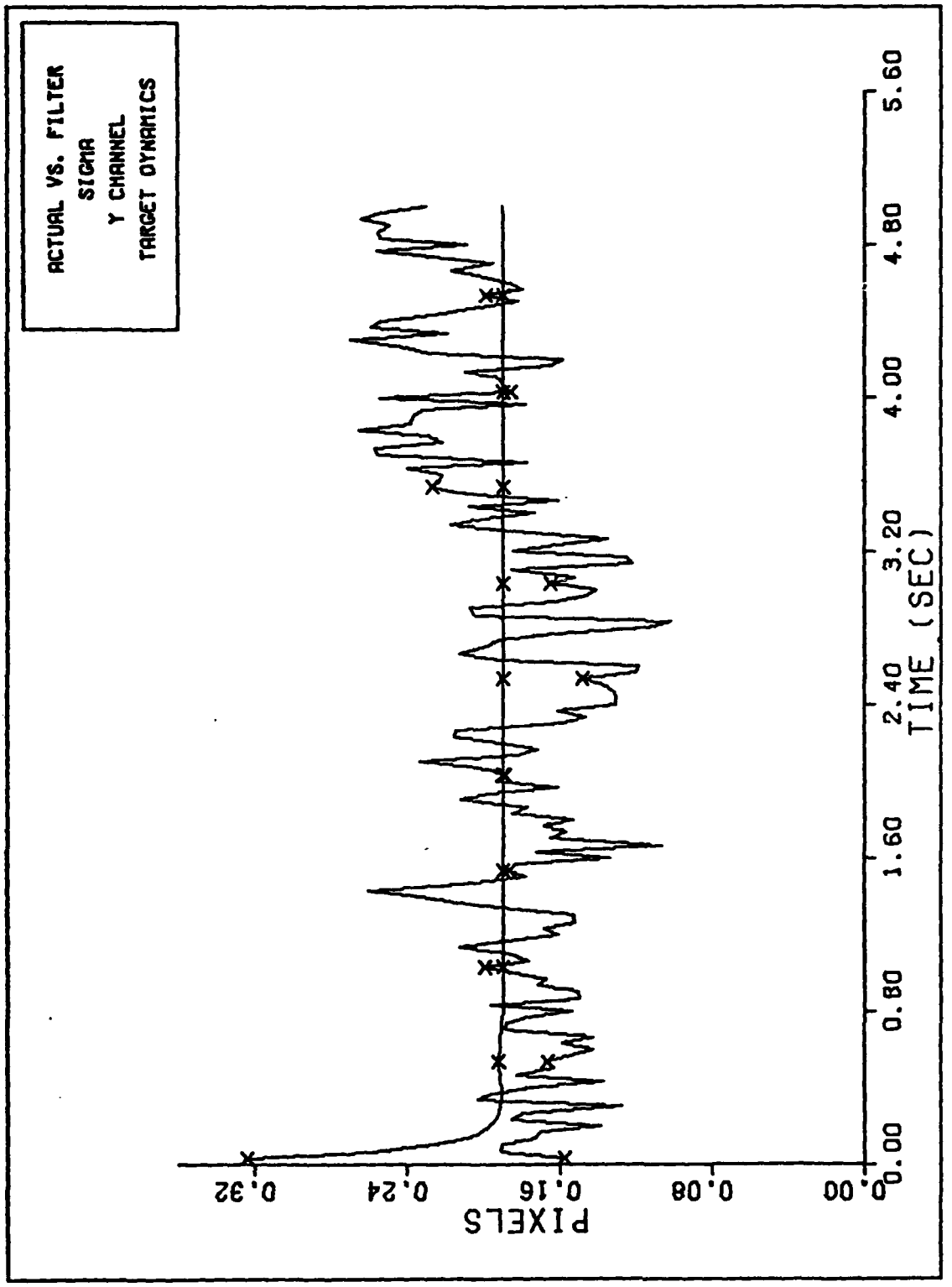
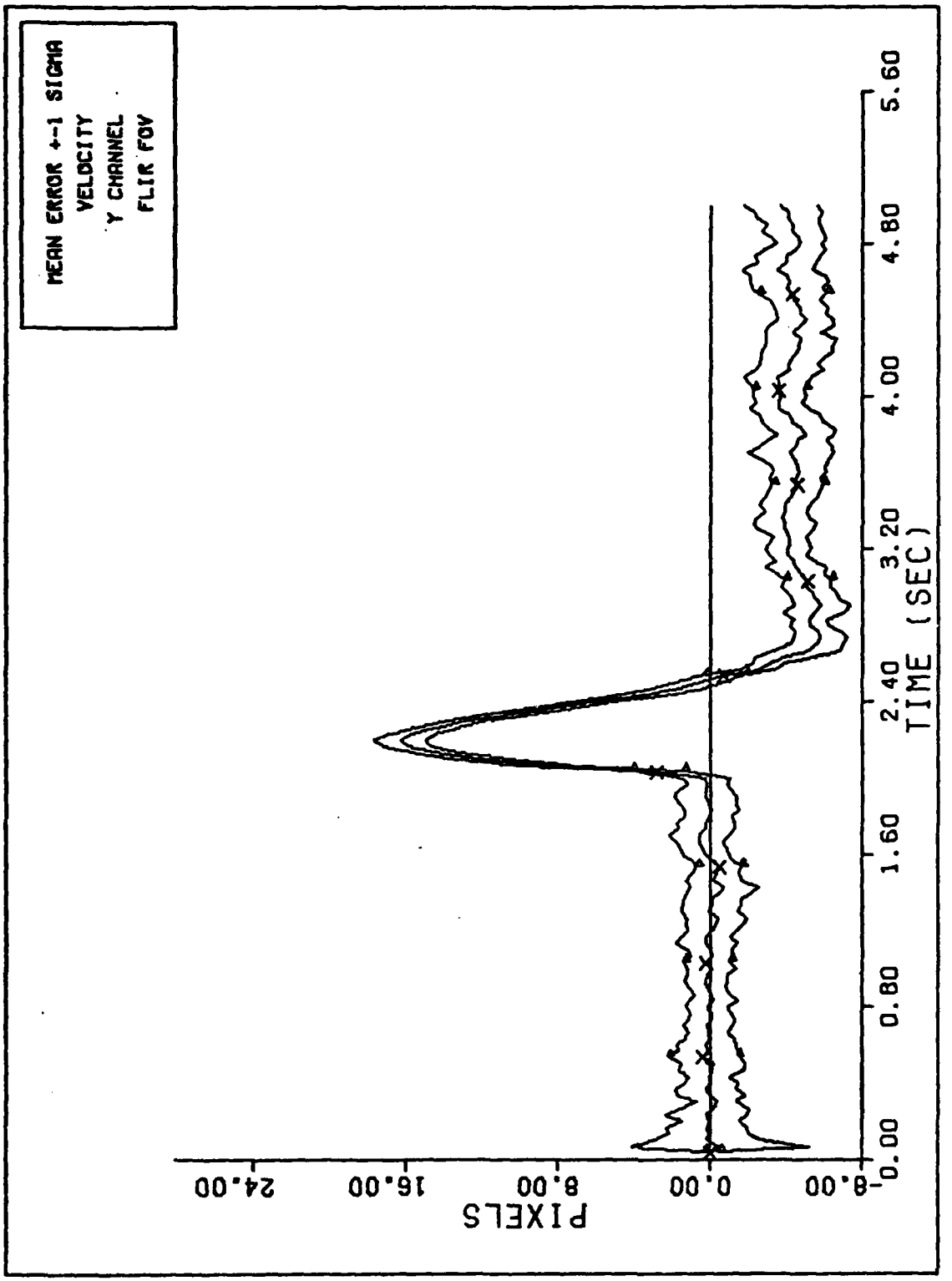


Figure C-5 20 g Q=600 Performance Plot



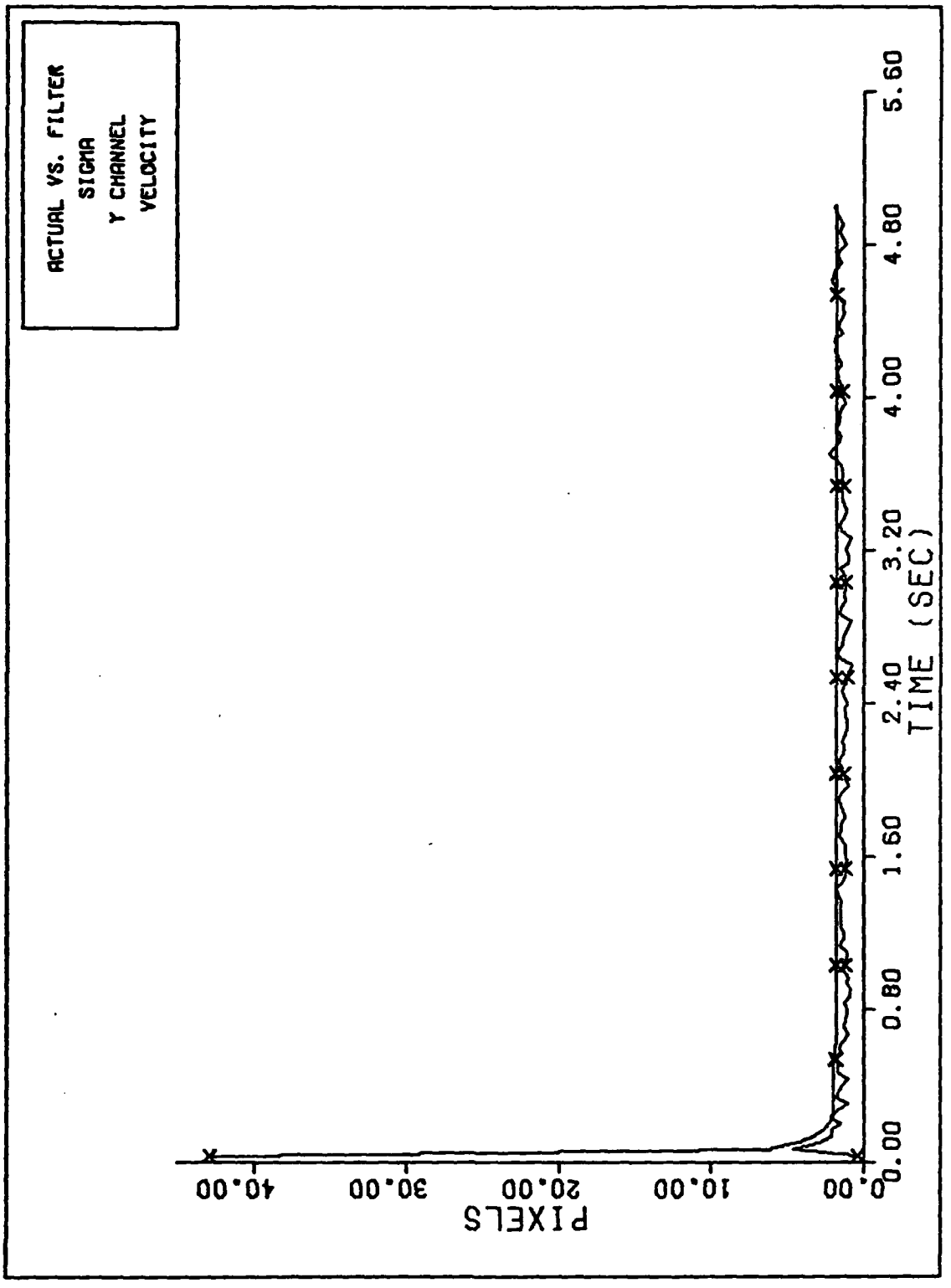
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-6 20 g Q=600 Performance Plot



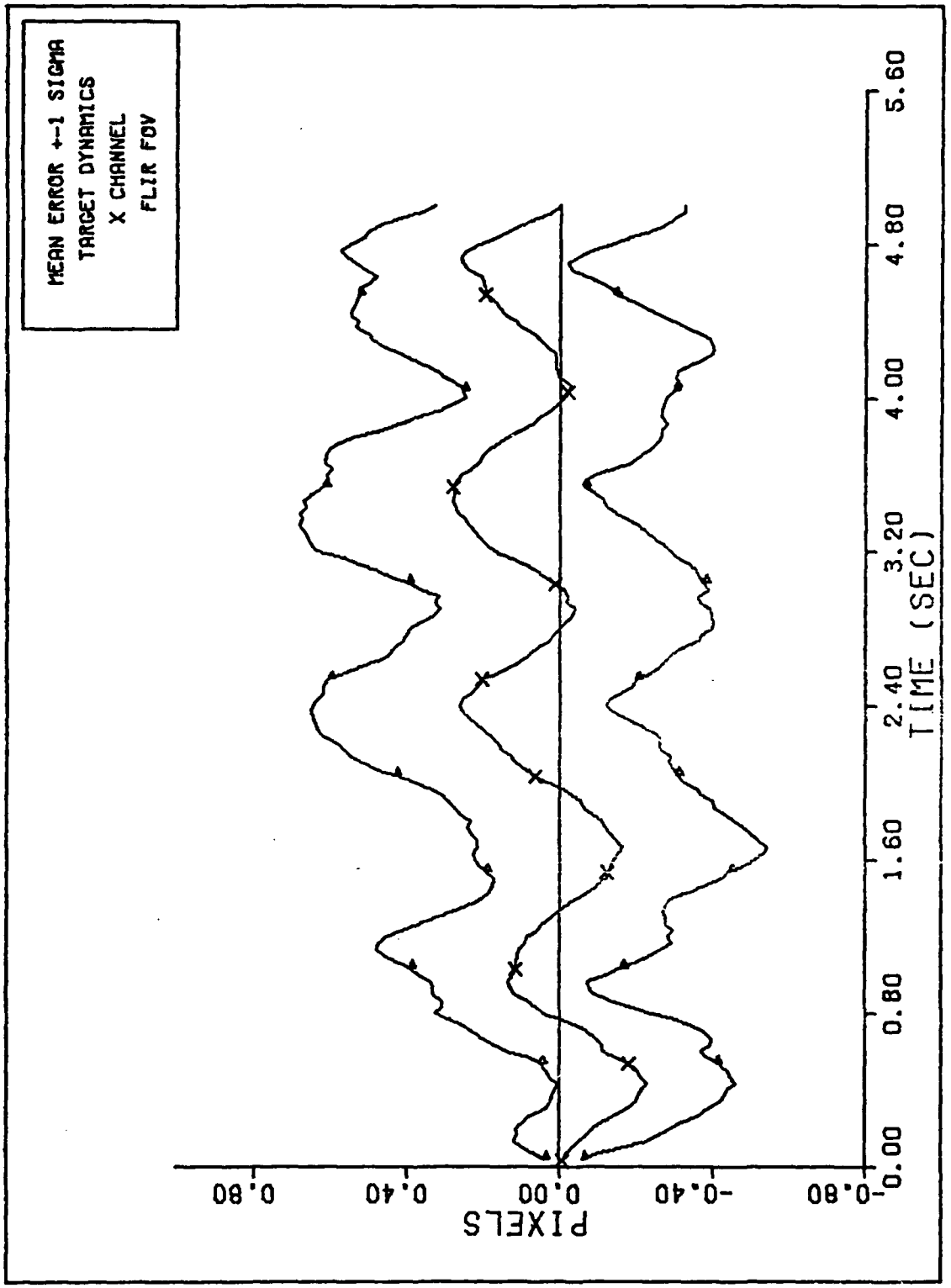
Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure C-7 20 g Q=600 Performance Plot



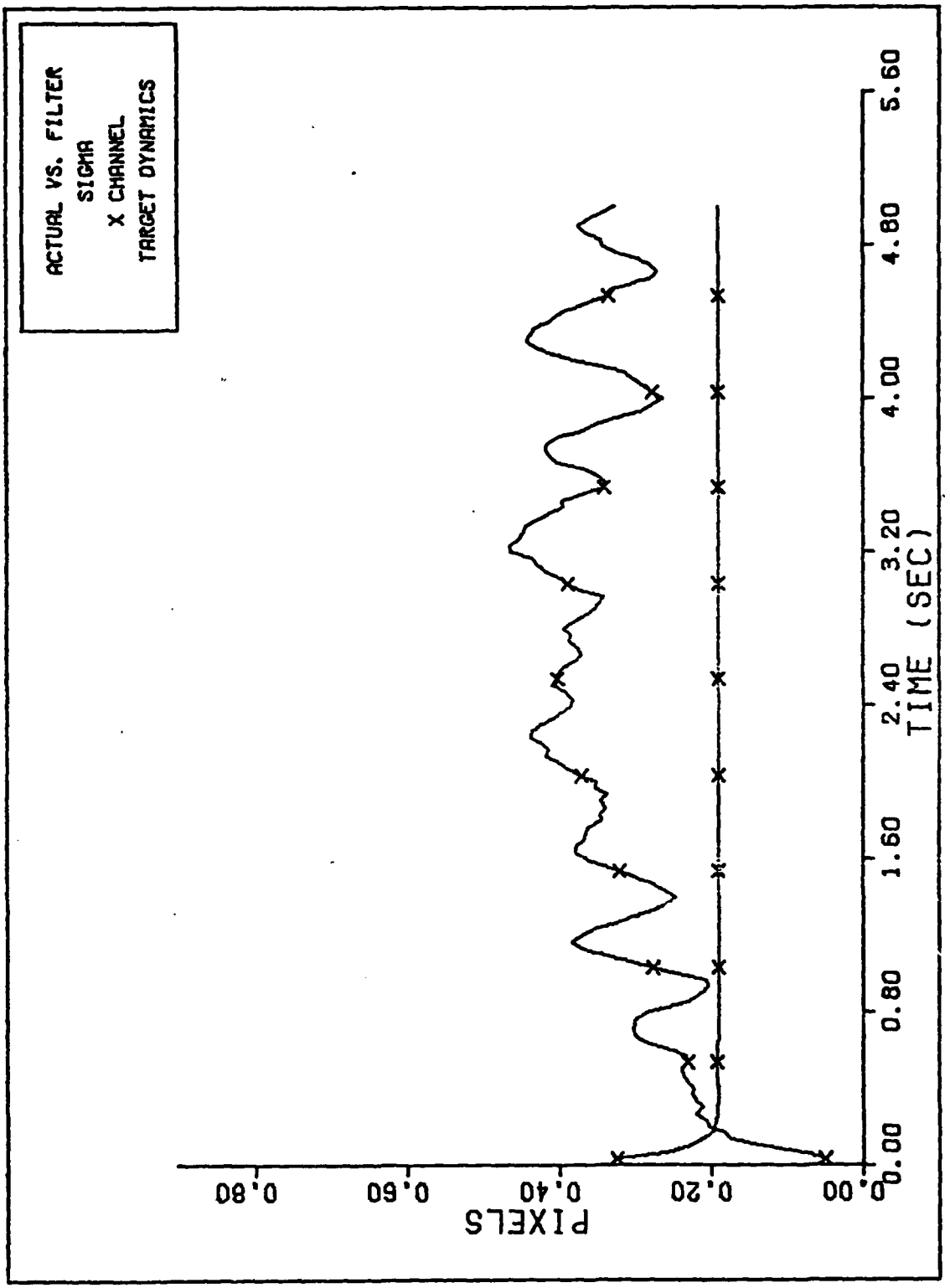
FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure C-8 20 g Q=600 Performance Plot



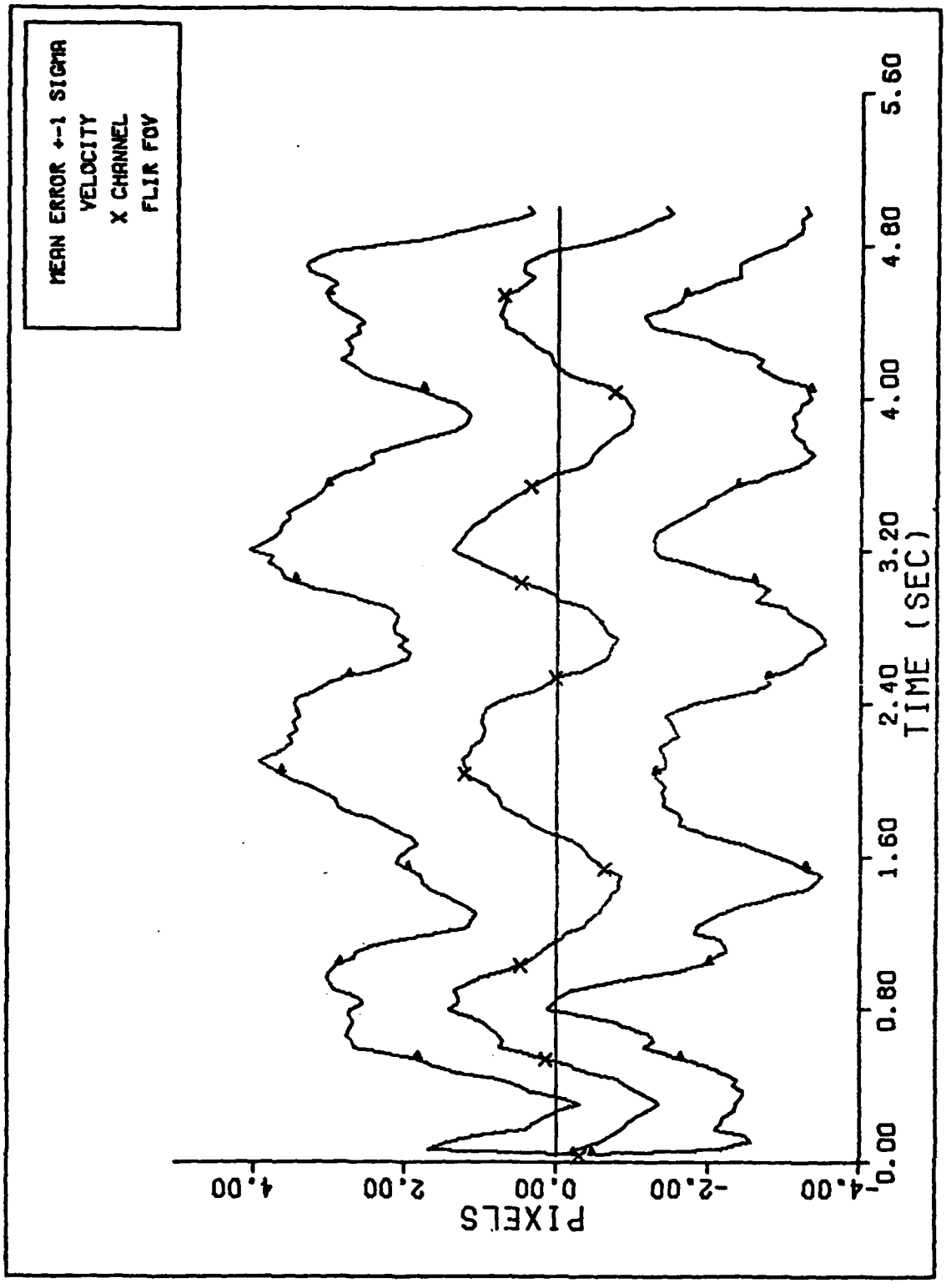
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure C-9 10 g Q=600 Performance Plot



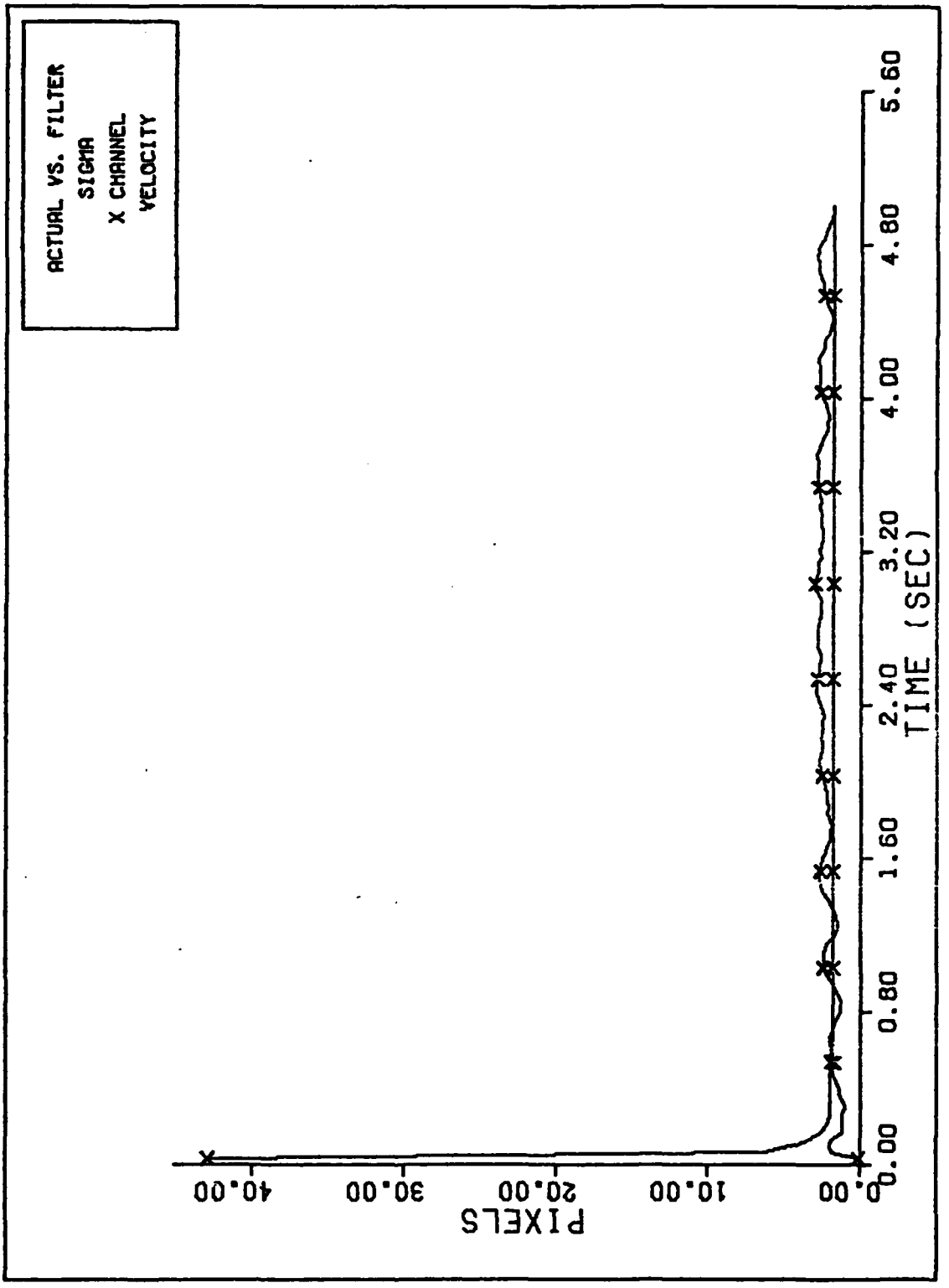
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-10 10 g Q=600 Performance Plot



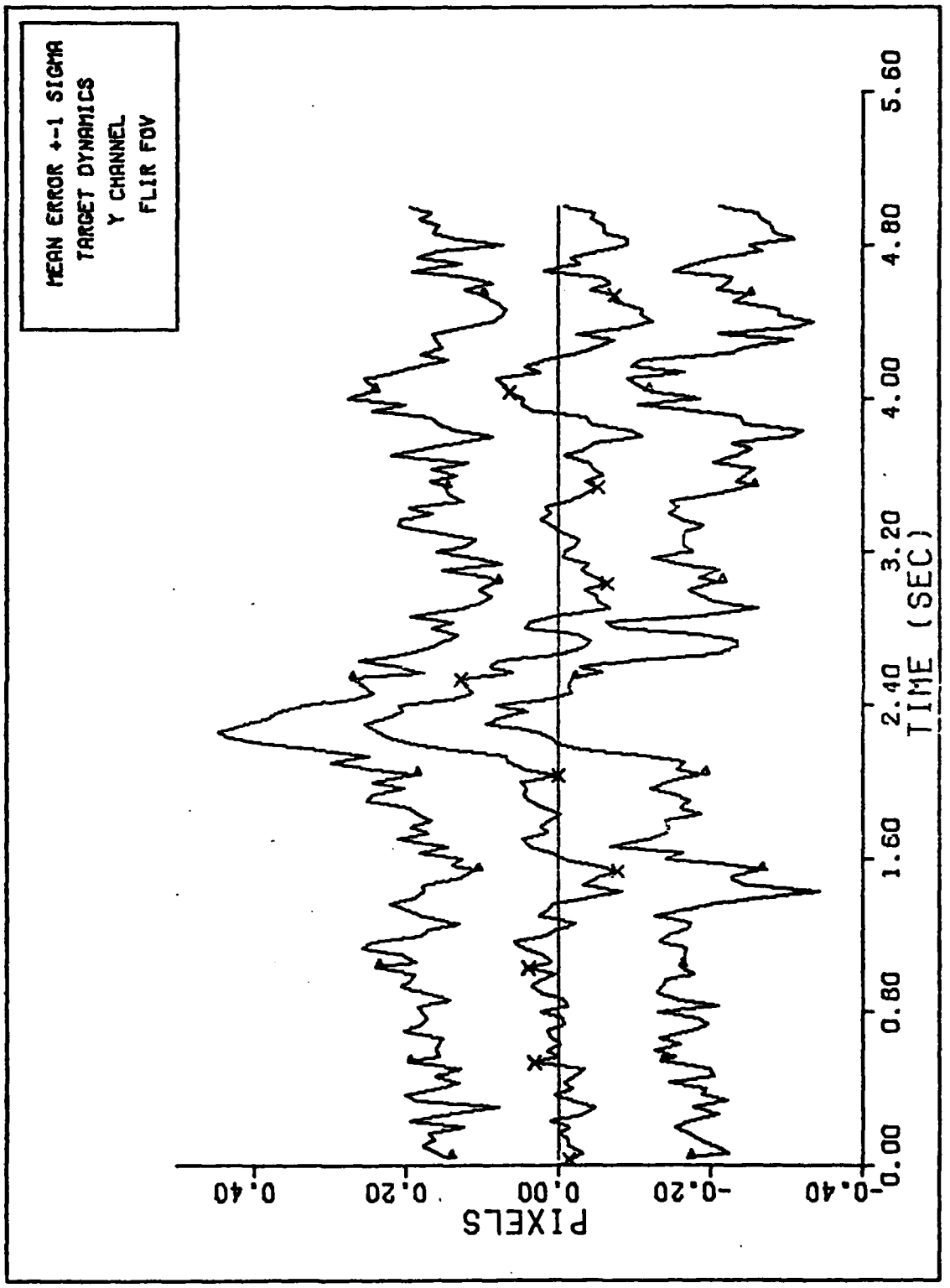
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure C-11 10 g Q=600 Performance Plot



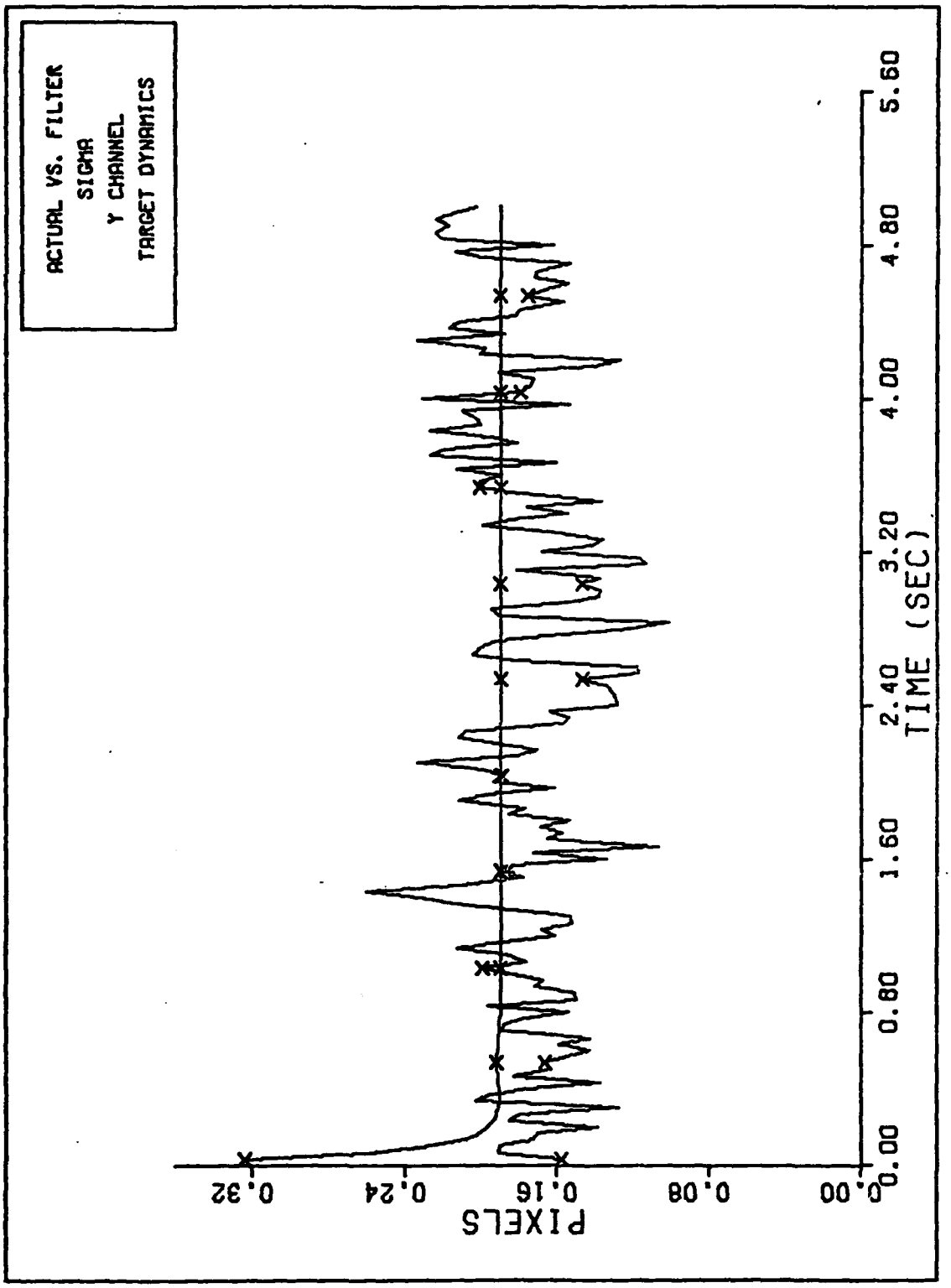
FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure C-12 10 g Q=600 Performance Plot



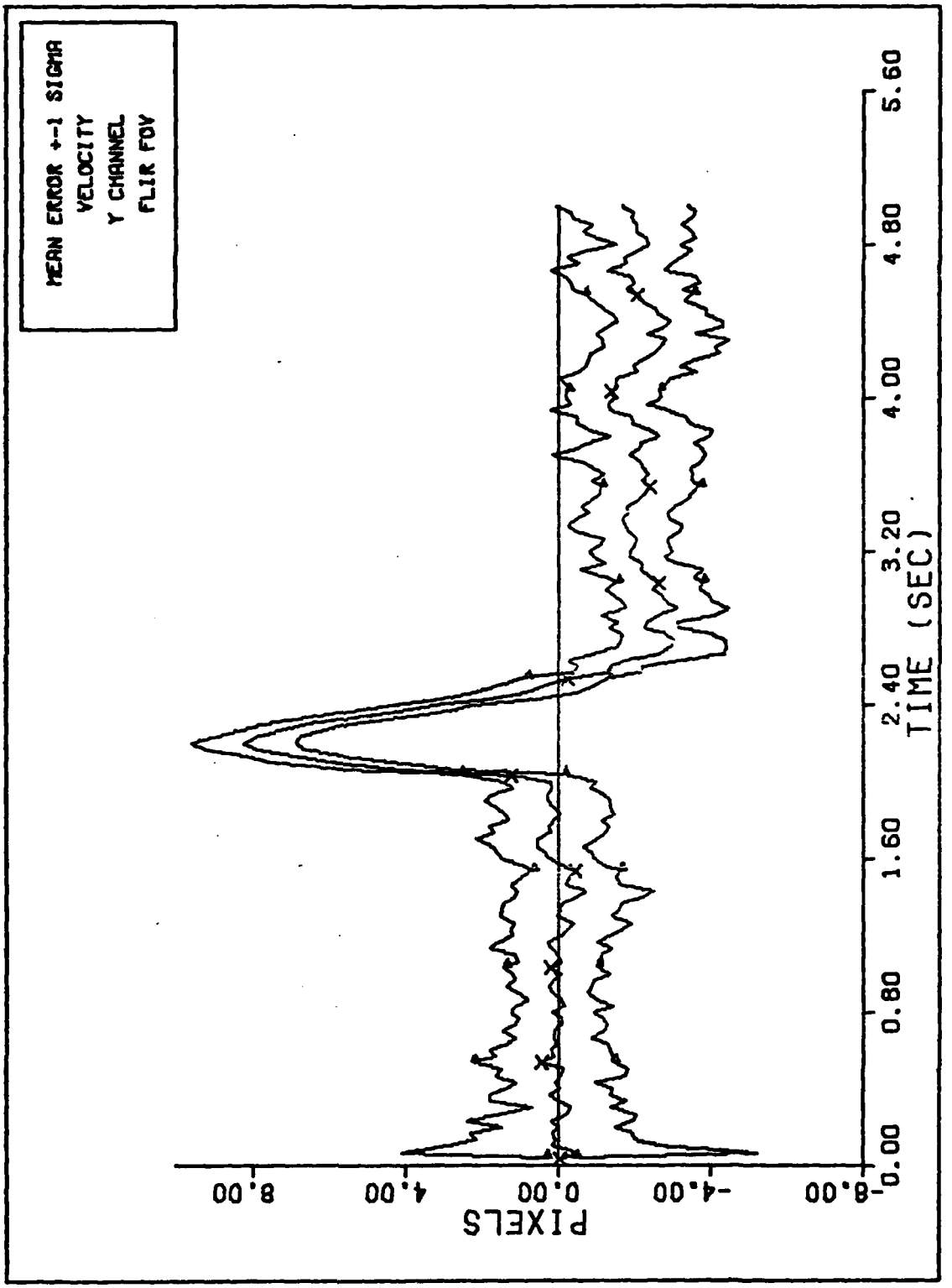
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure C-13 10 g Q=600 Performance Plot



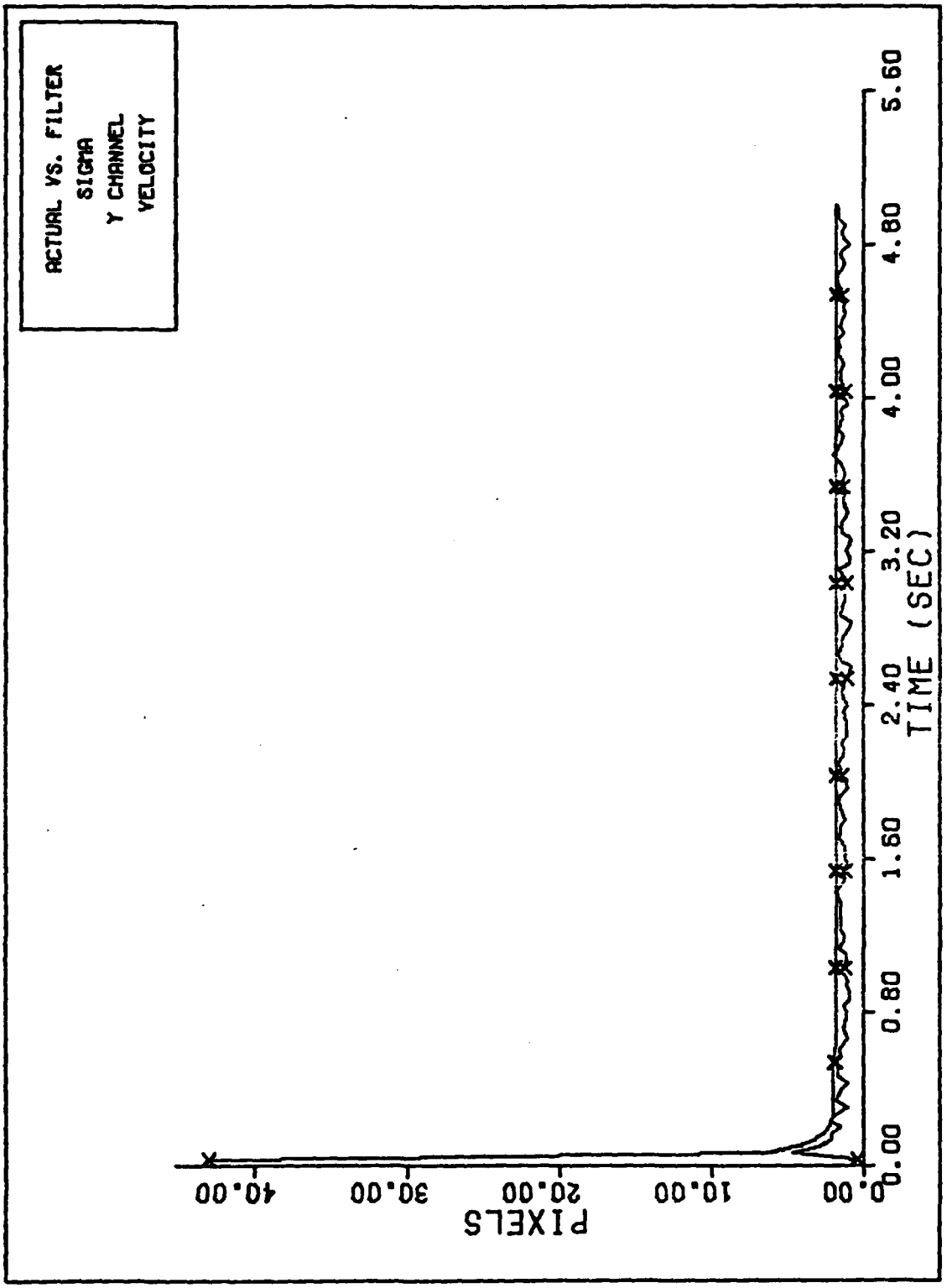
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-14 10 g Q=600 Performance Plot



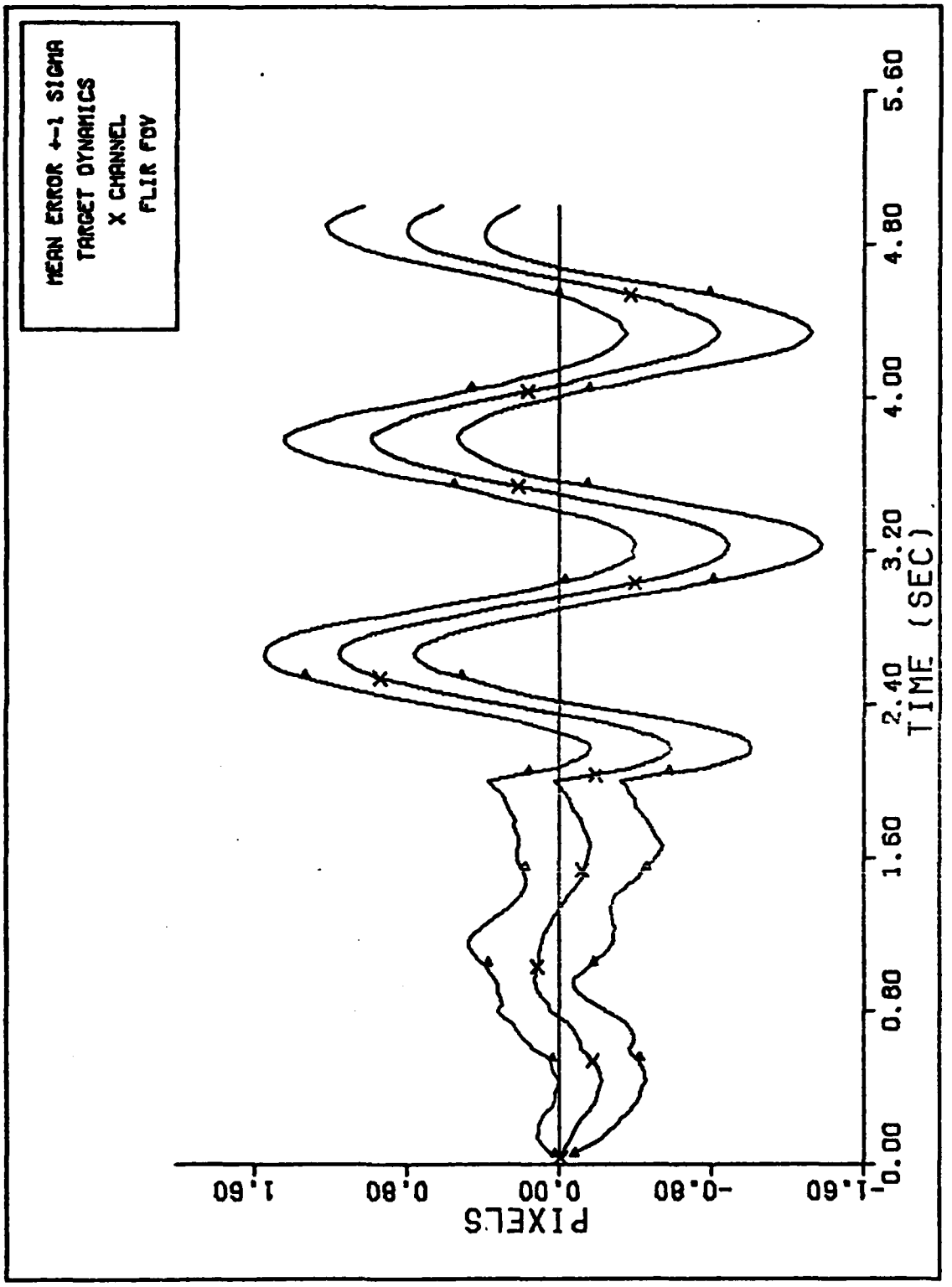
Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure C-15 10 g Q=600 Performance Plot



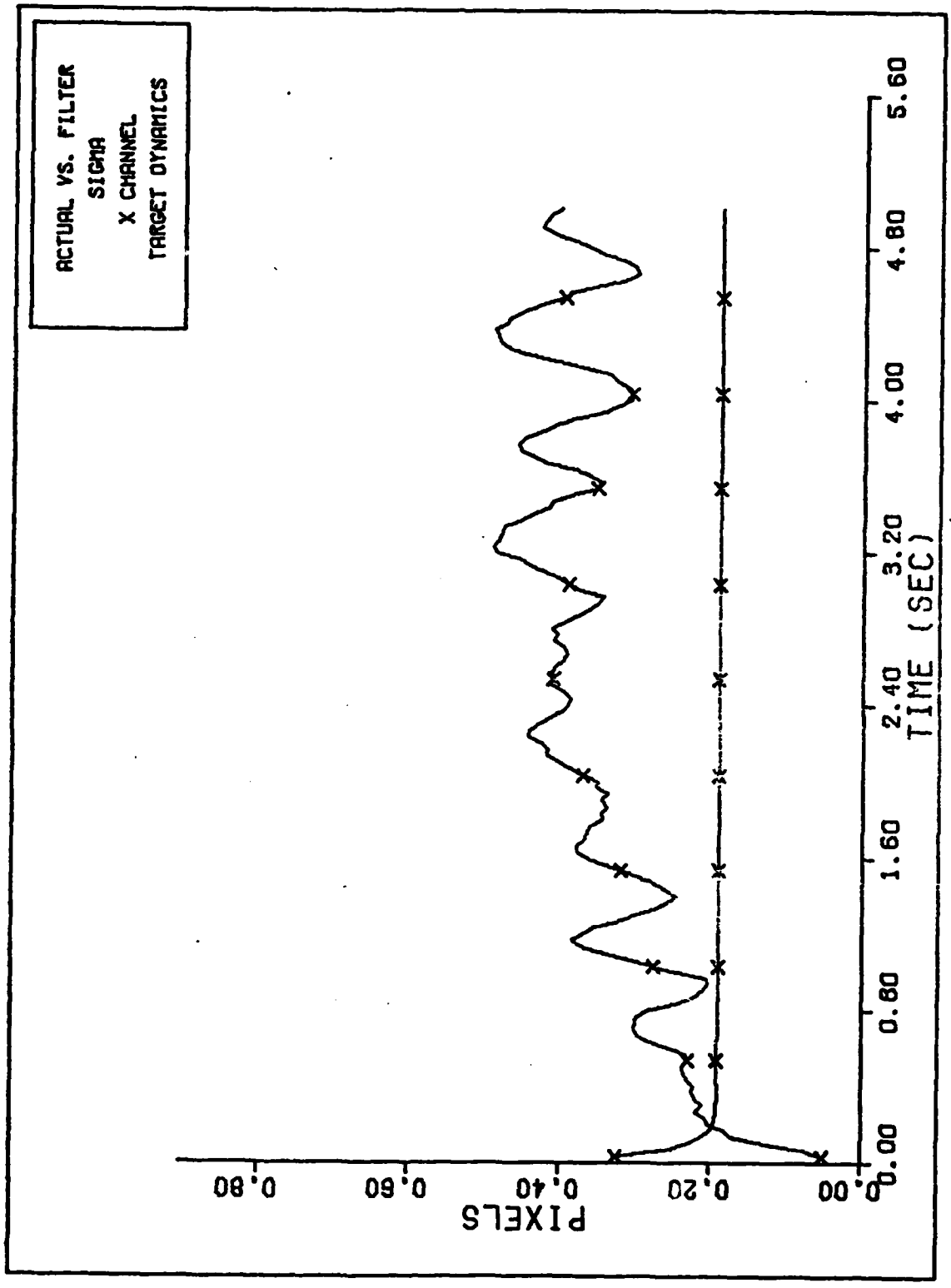
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-16 10 g Q=600 Performance Plot



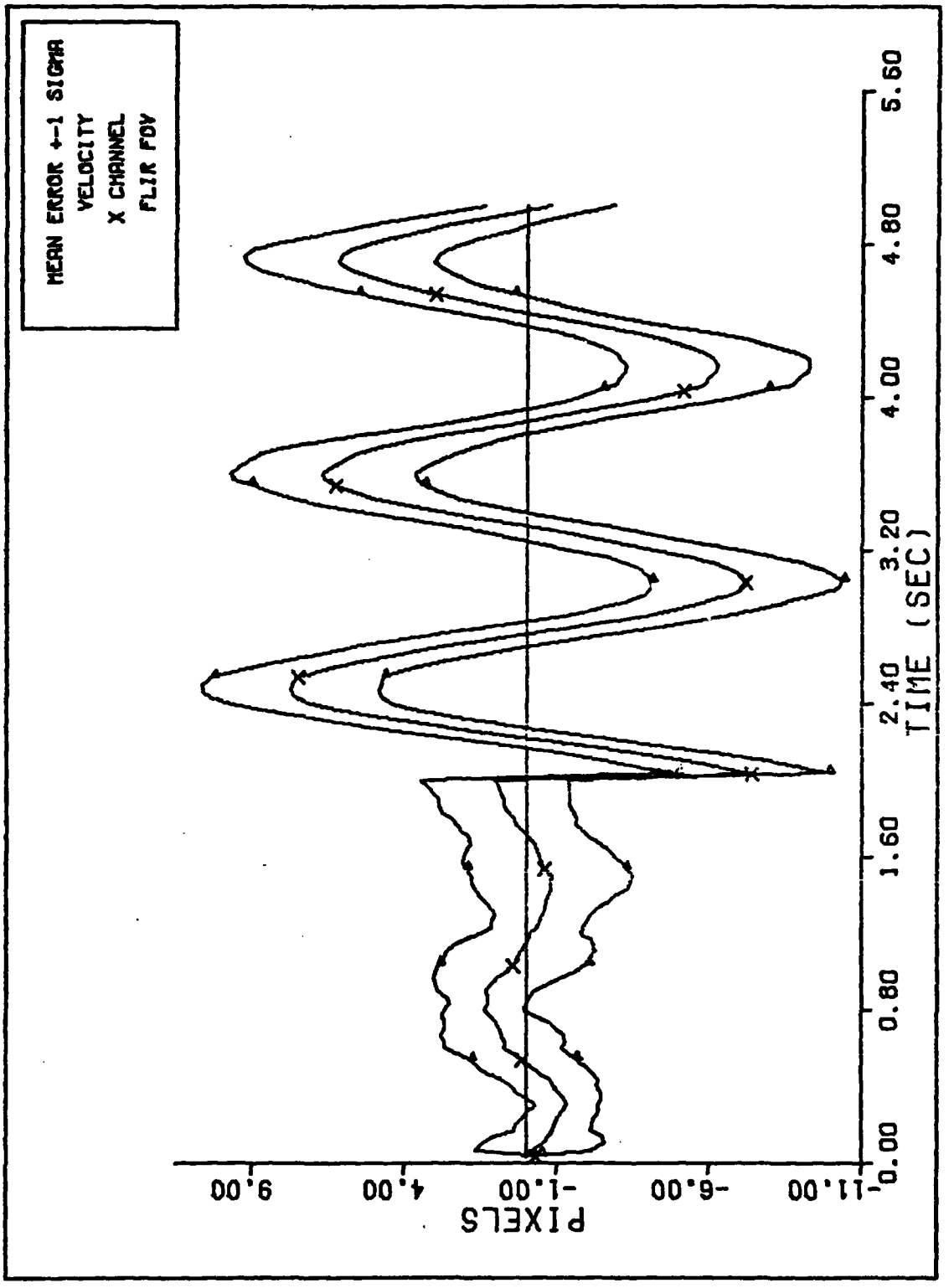
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure C-17 2 g Q=600 Performance Plot



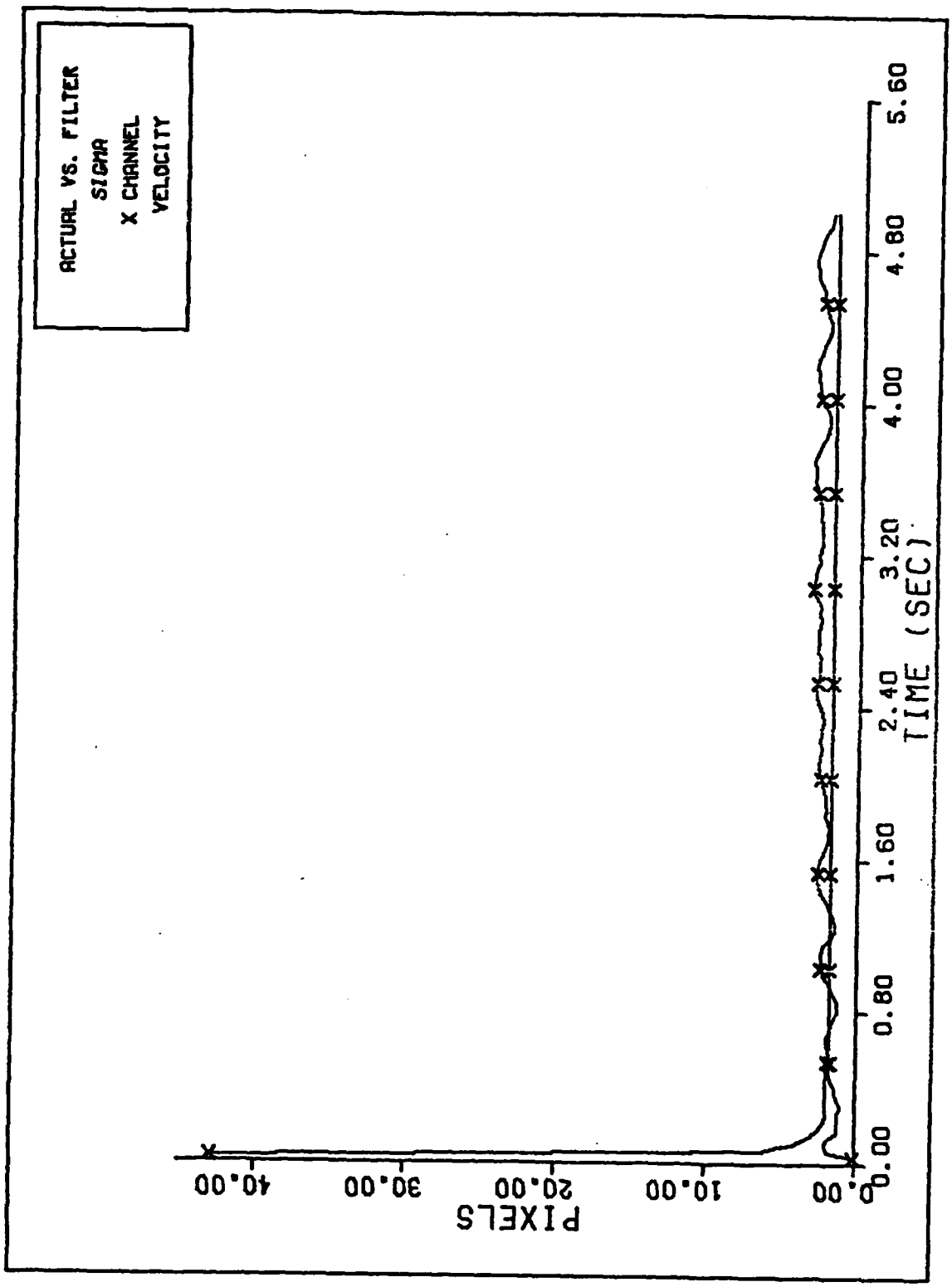
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-18 2 g Q=600 Performance Plot



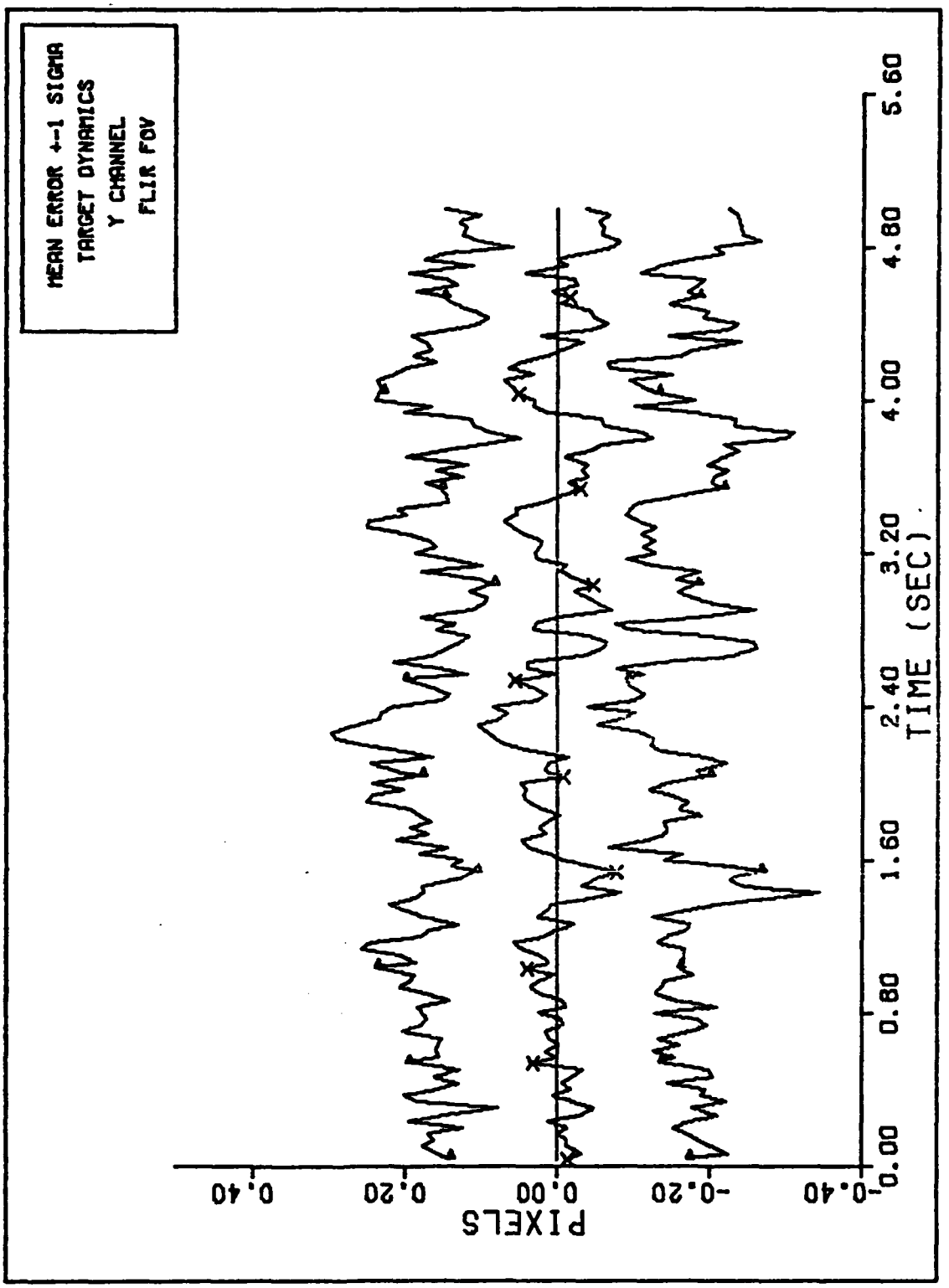
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure C-19 2 g Q=600 Performance Plot



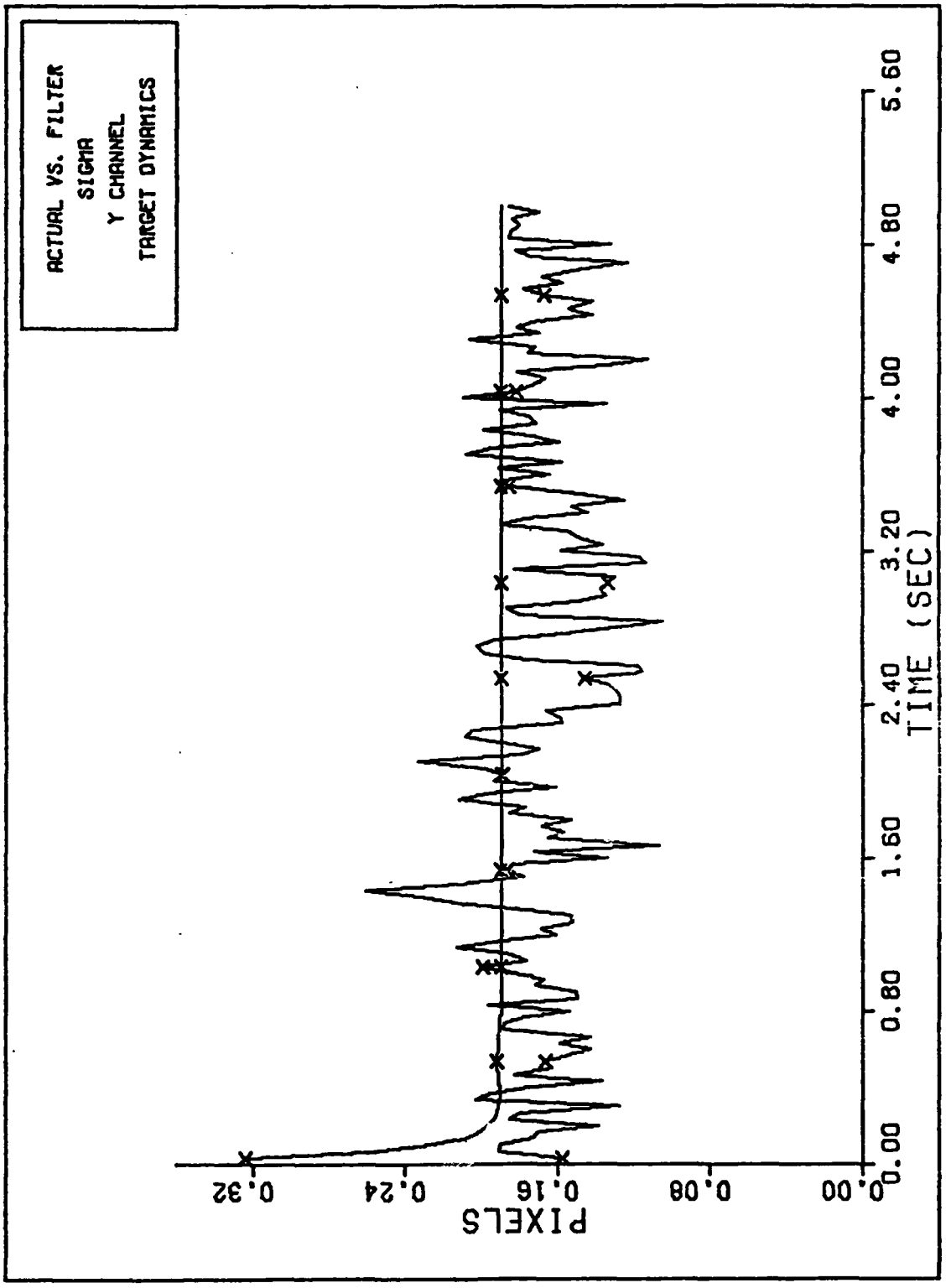
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-20 2 g Q=600 Performance Plot



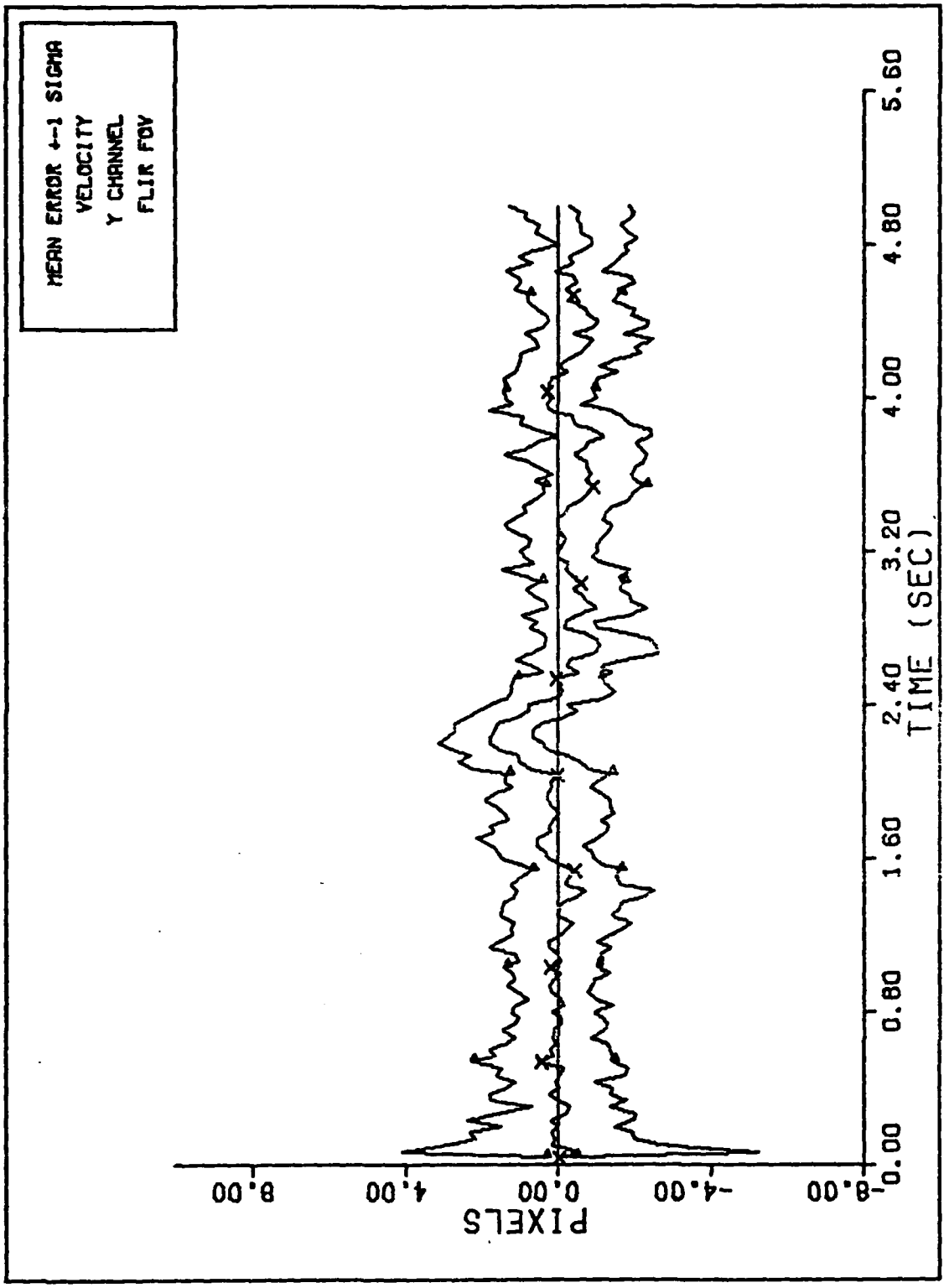
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure C-21 2 g Q=600 Performance Plot



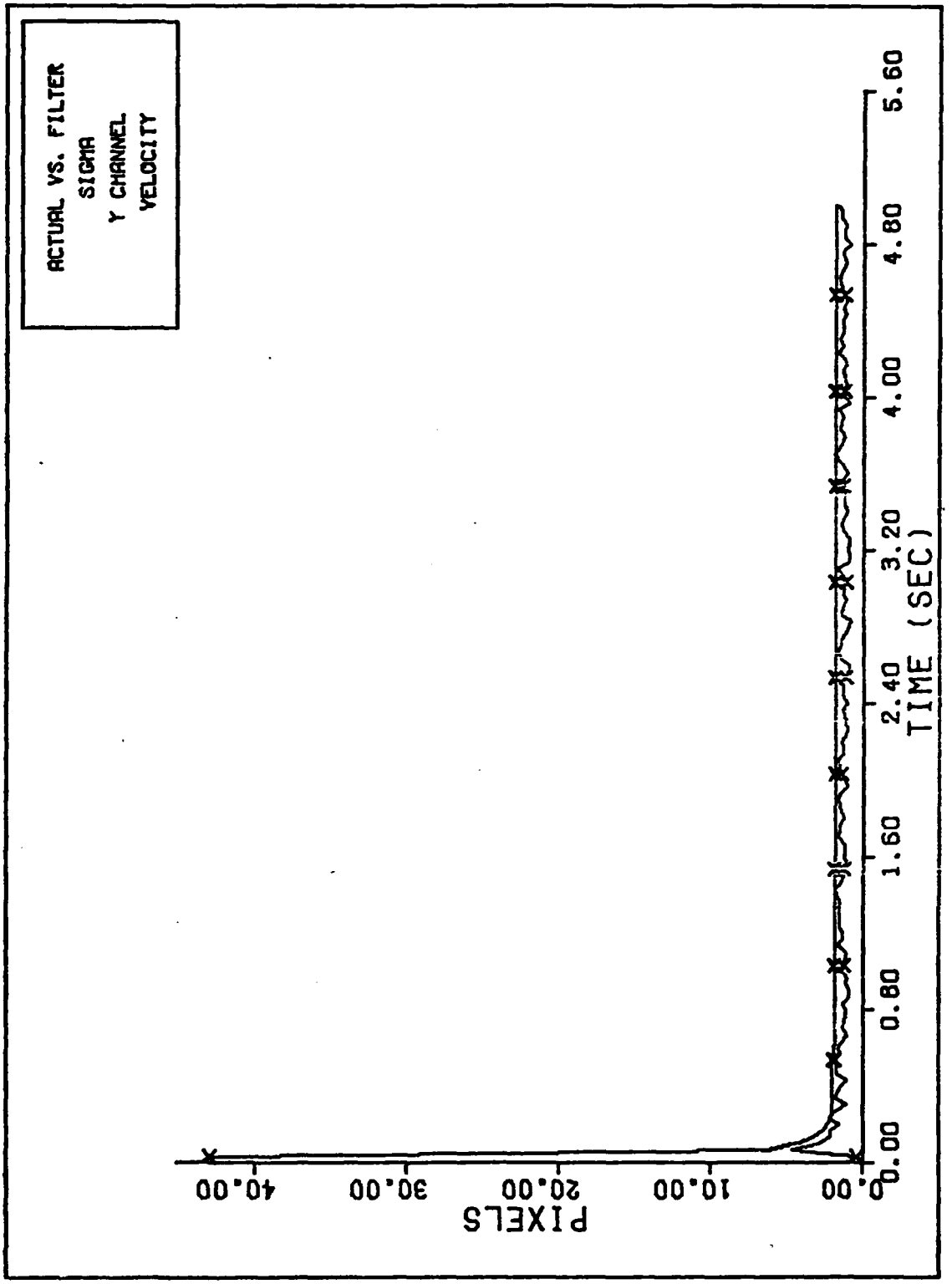
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-22 2 g Q=600 Performance Plot



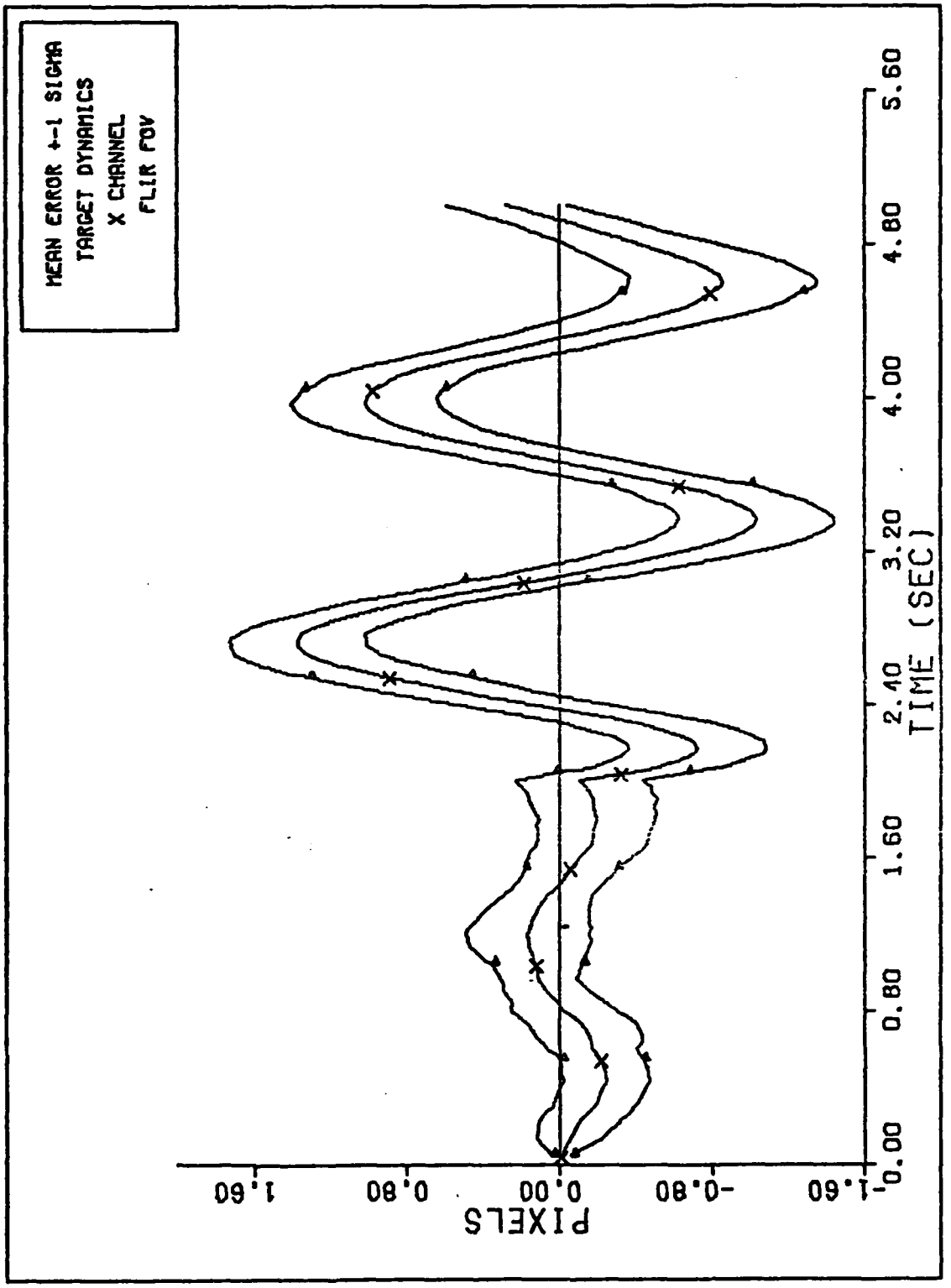
Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure C-23 2 g Q=600 Performance Plot



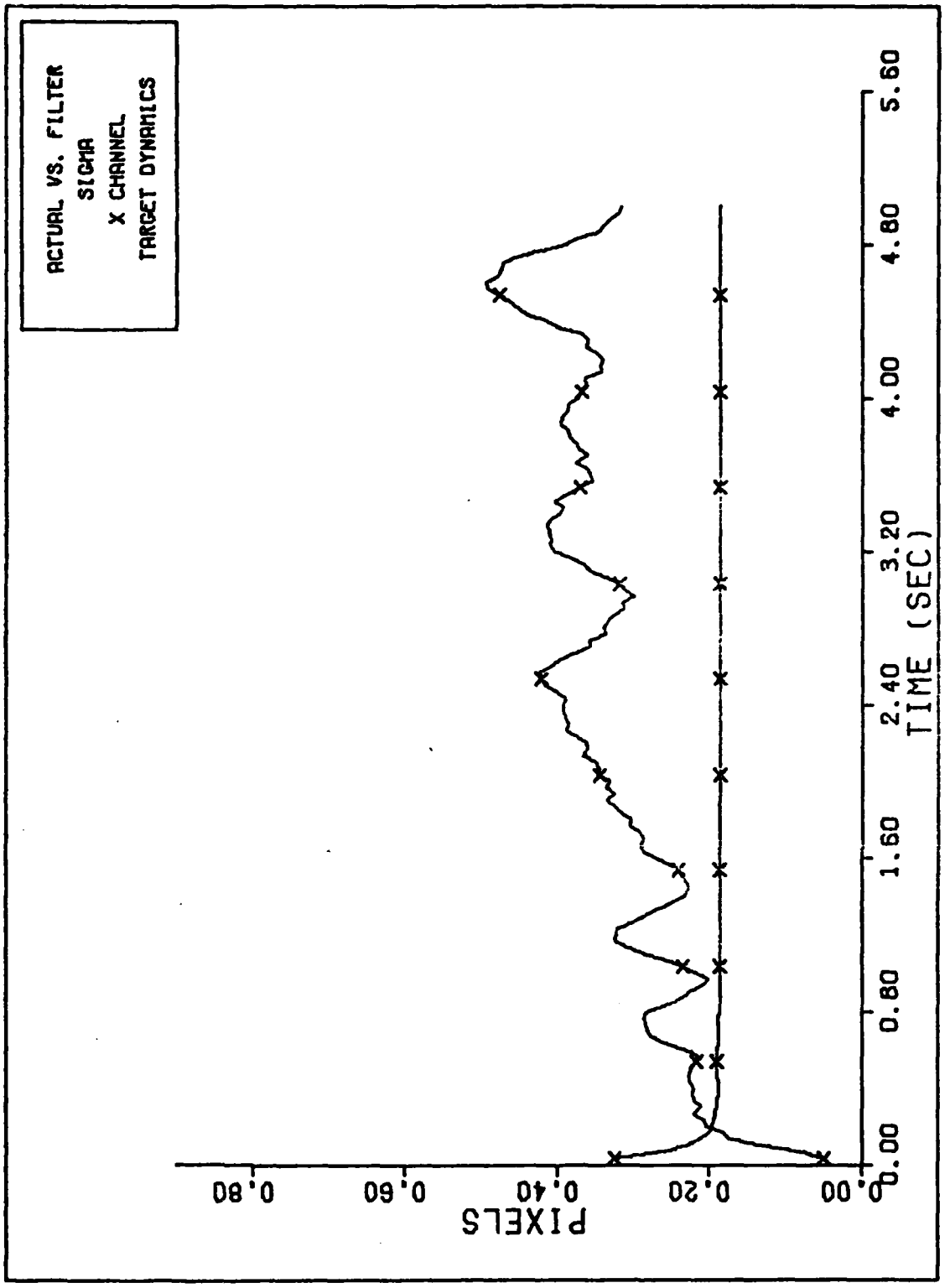
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-24 2 g Q=600 Performance Plot



X CHANNEL DYNAMICS ERROR (S/N= 12.5)

Figure C-25 2 g Q=150 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure C-26 2 g Q=150 Performance Plot

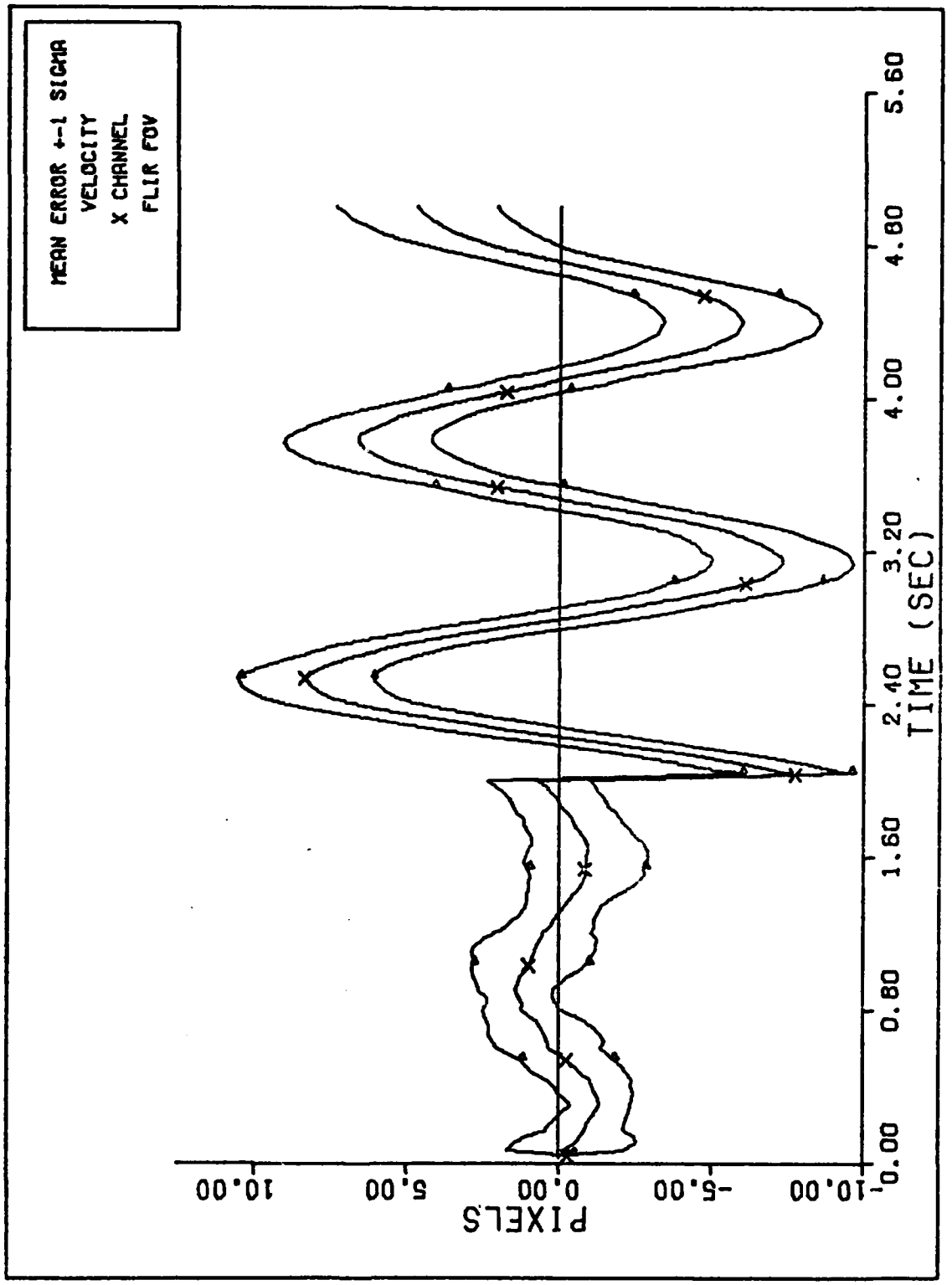
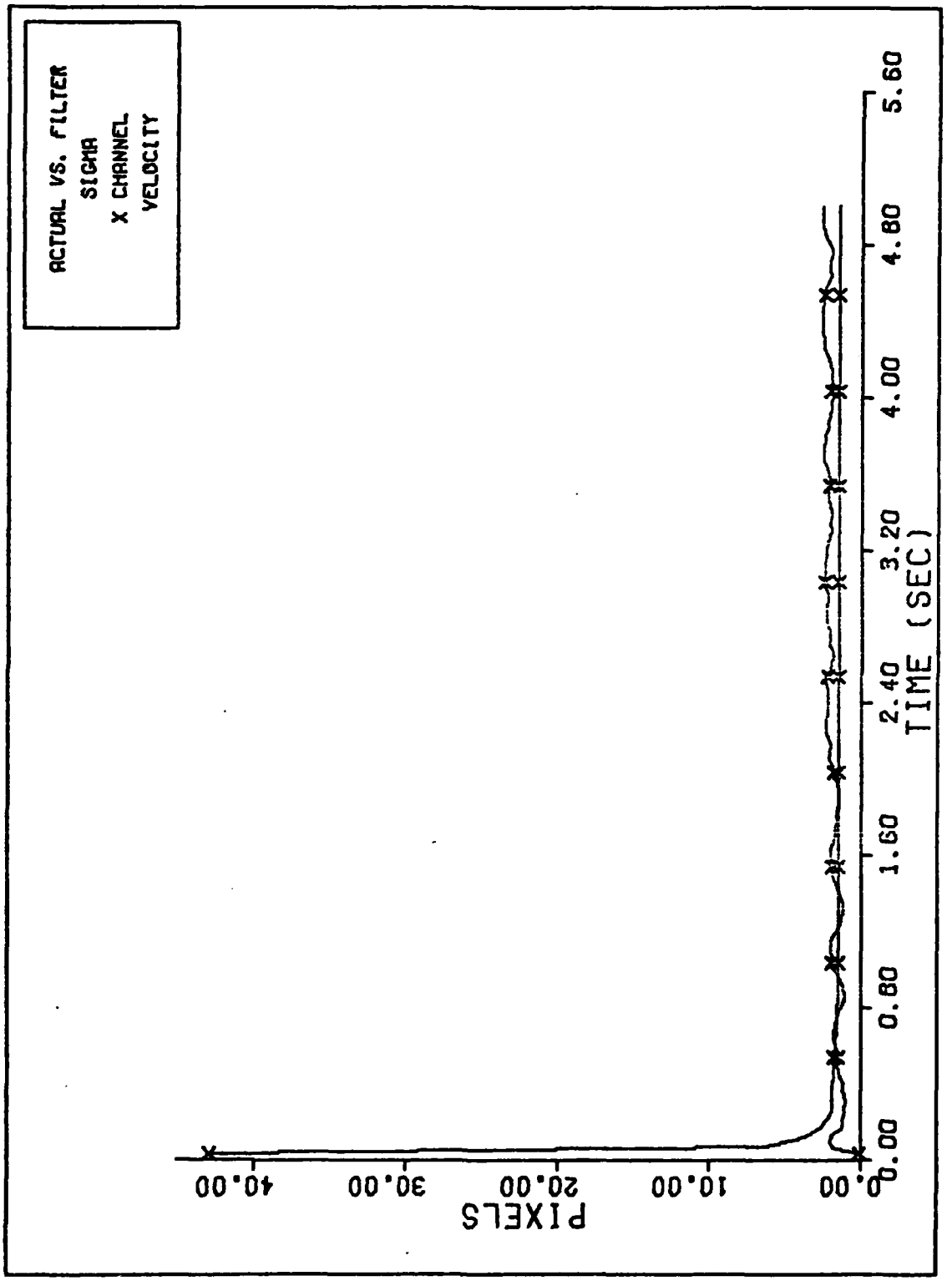


Figure C-27 2 g Q=150 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure C-28 2 g Q=150 Performance Plot

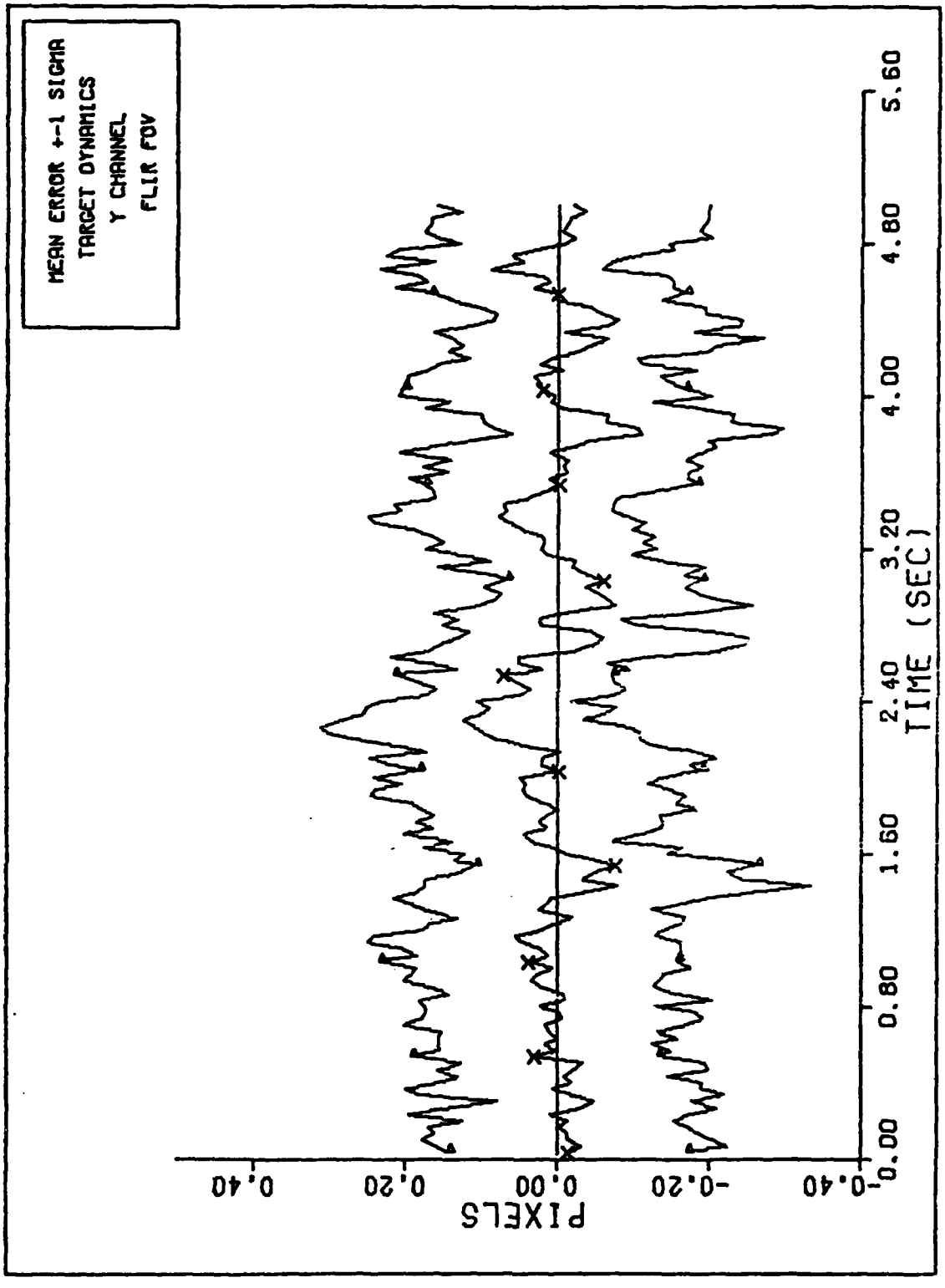
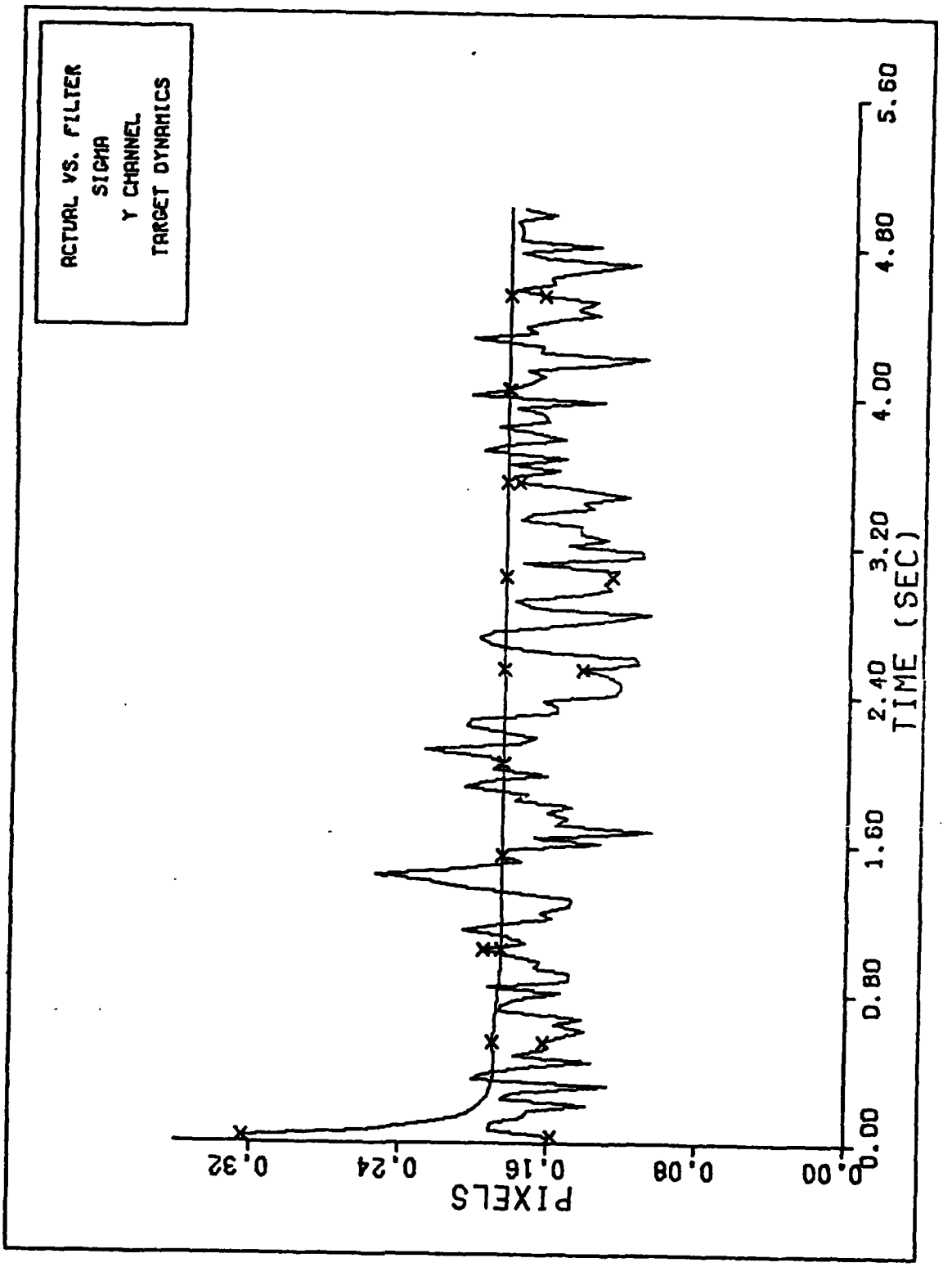
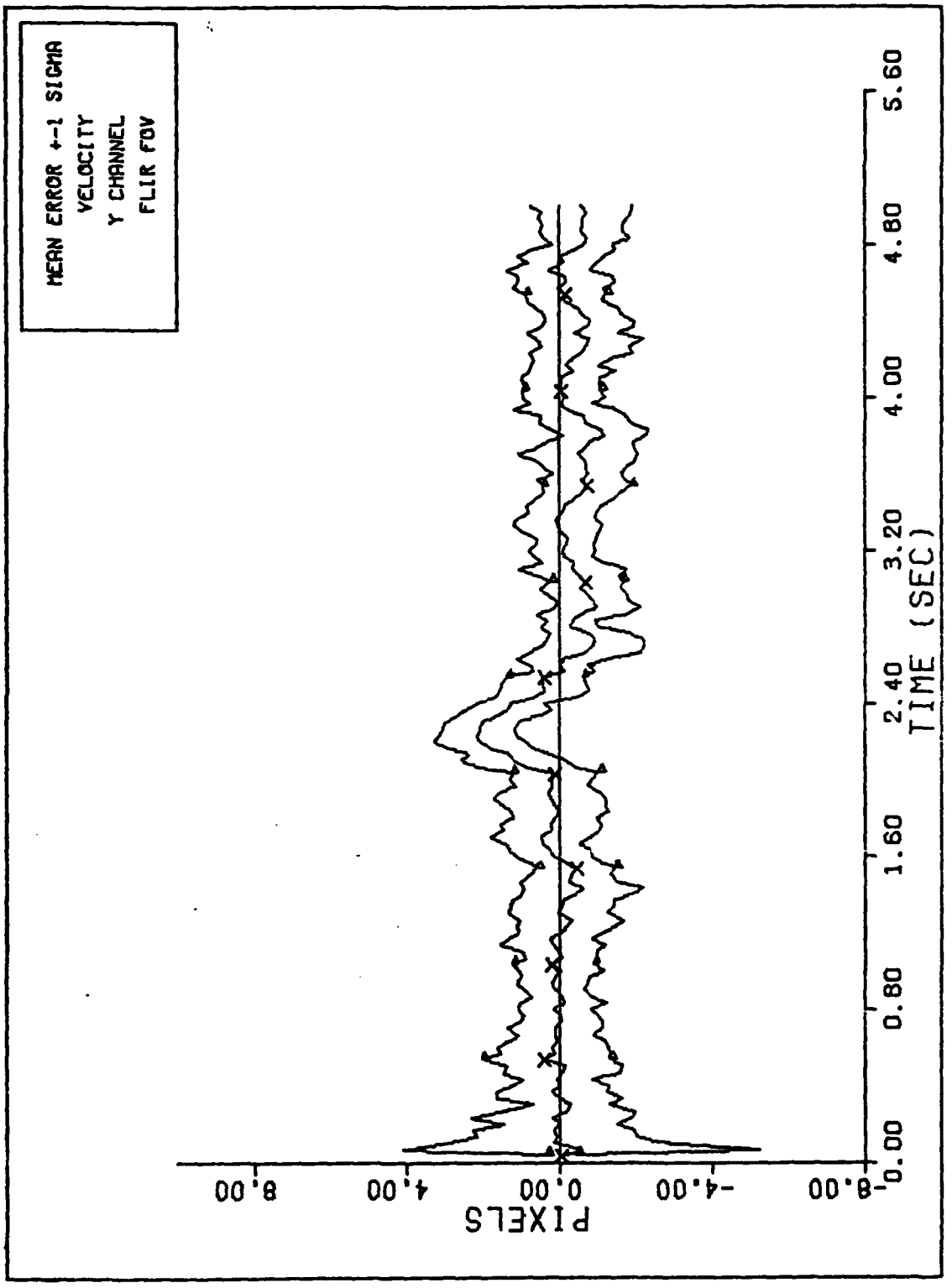


Figure C-29 2 g Q=150 Performance Plot



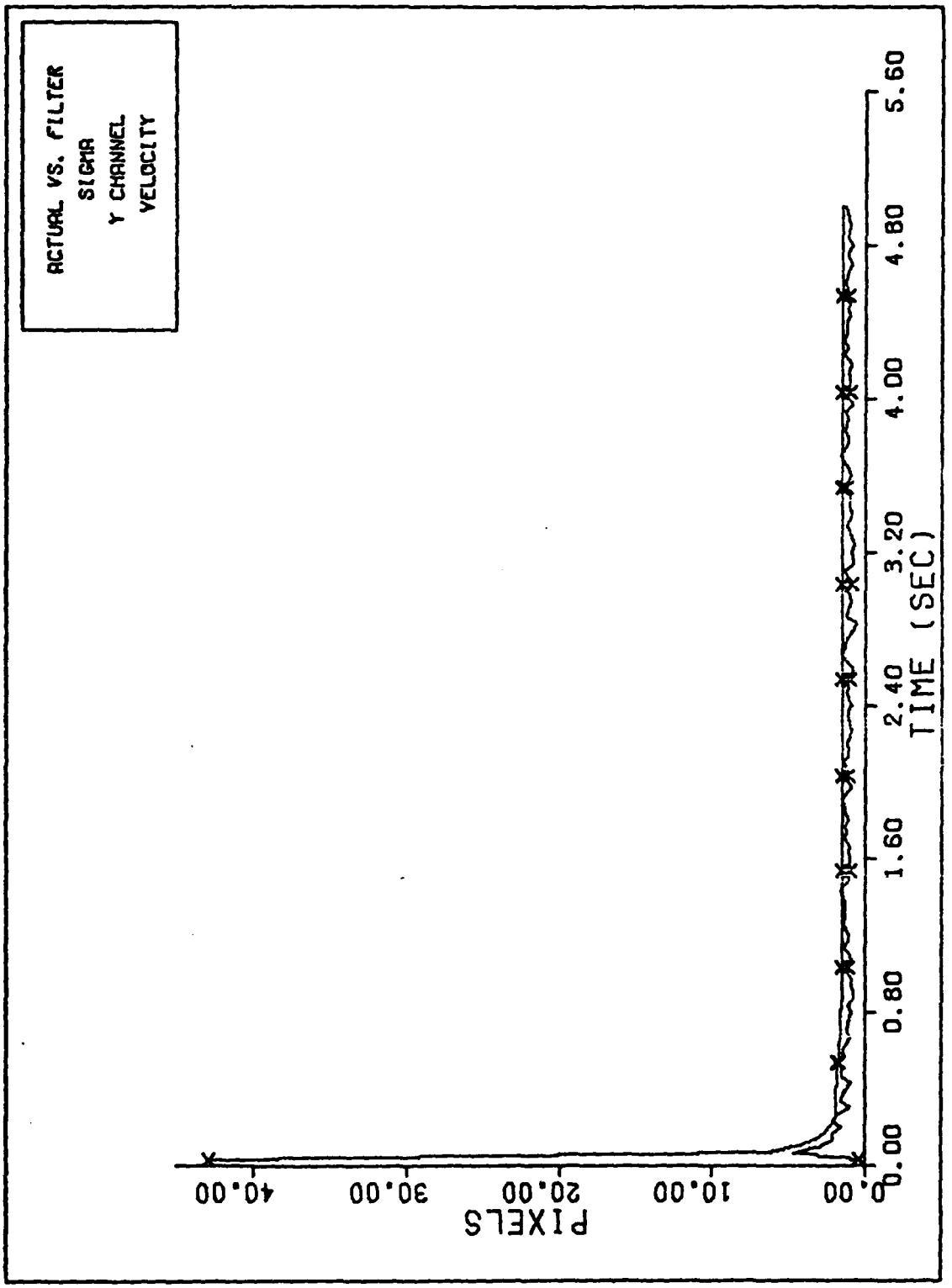
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure C-30 2 g Q=150 Performance Plot



Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure C-31 2 g Q=150 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure C-32 2 g Q=150 Performance Plot

APPENDIX D

Performance Plots for the Constant Turn Rate Filter

This appendix contains the plotted outputs of the performance of the BH filter. Three trajectories were simulated and eight plots are included for each case. They were:

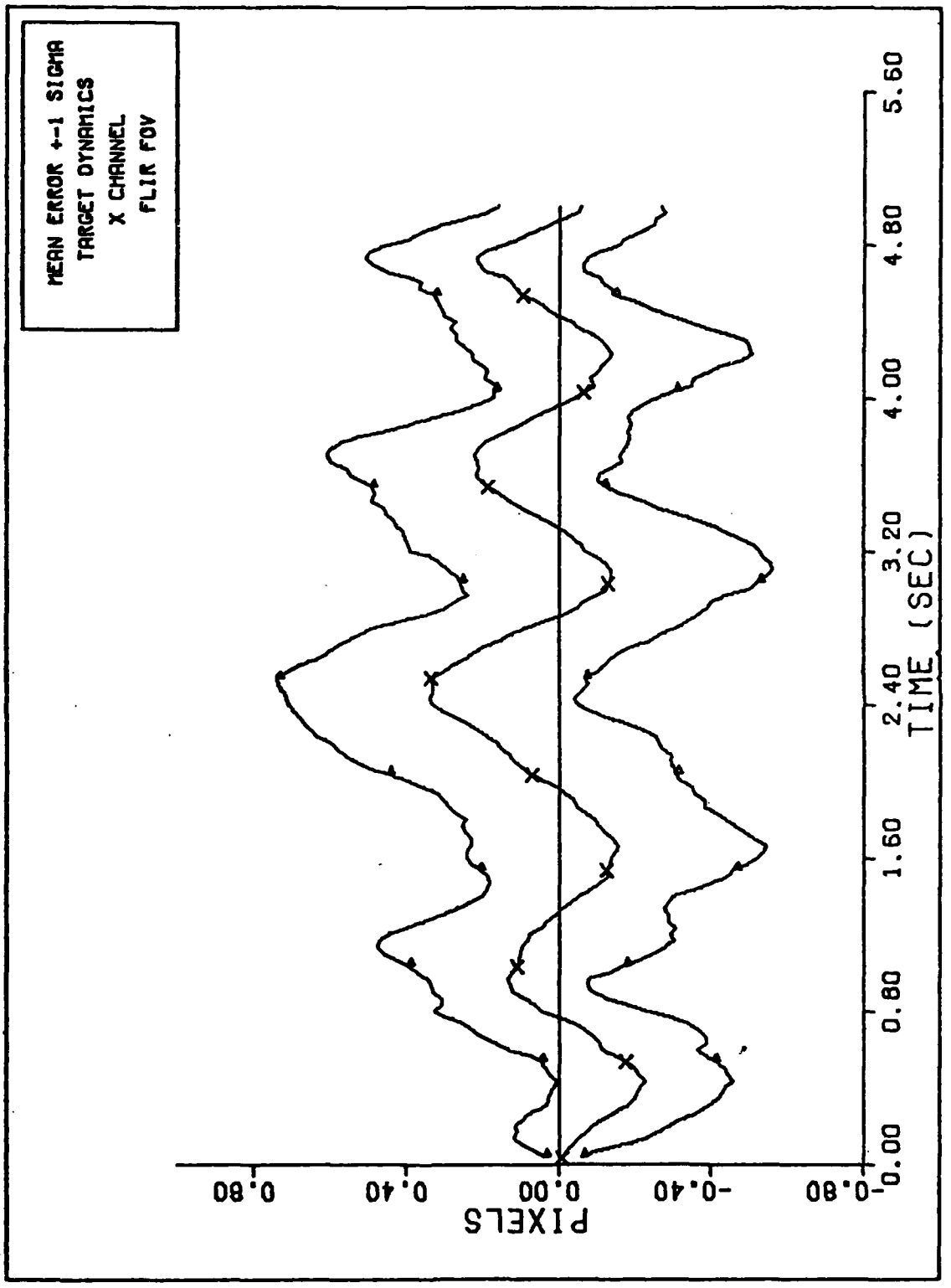
--mean error $\pm$ 1 sigma of the filter estimate of the 'x' and 'y' target position. These plots were used for checking the mean bias error and rms position error in general.

--mean error $\pm$ 1 sigma of the filter estimate of the 'x' and 'y' target velocity. These plots were used to check any mean bias errors and rms velocity errors in general.

--true and filter-indicated standard deviation of the 'x' and 'y' position. These plots were used in tuning the filters to the various trajectories simulated.

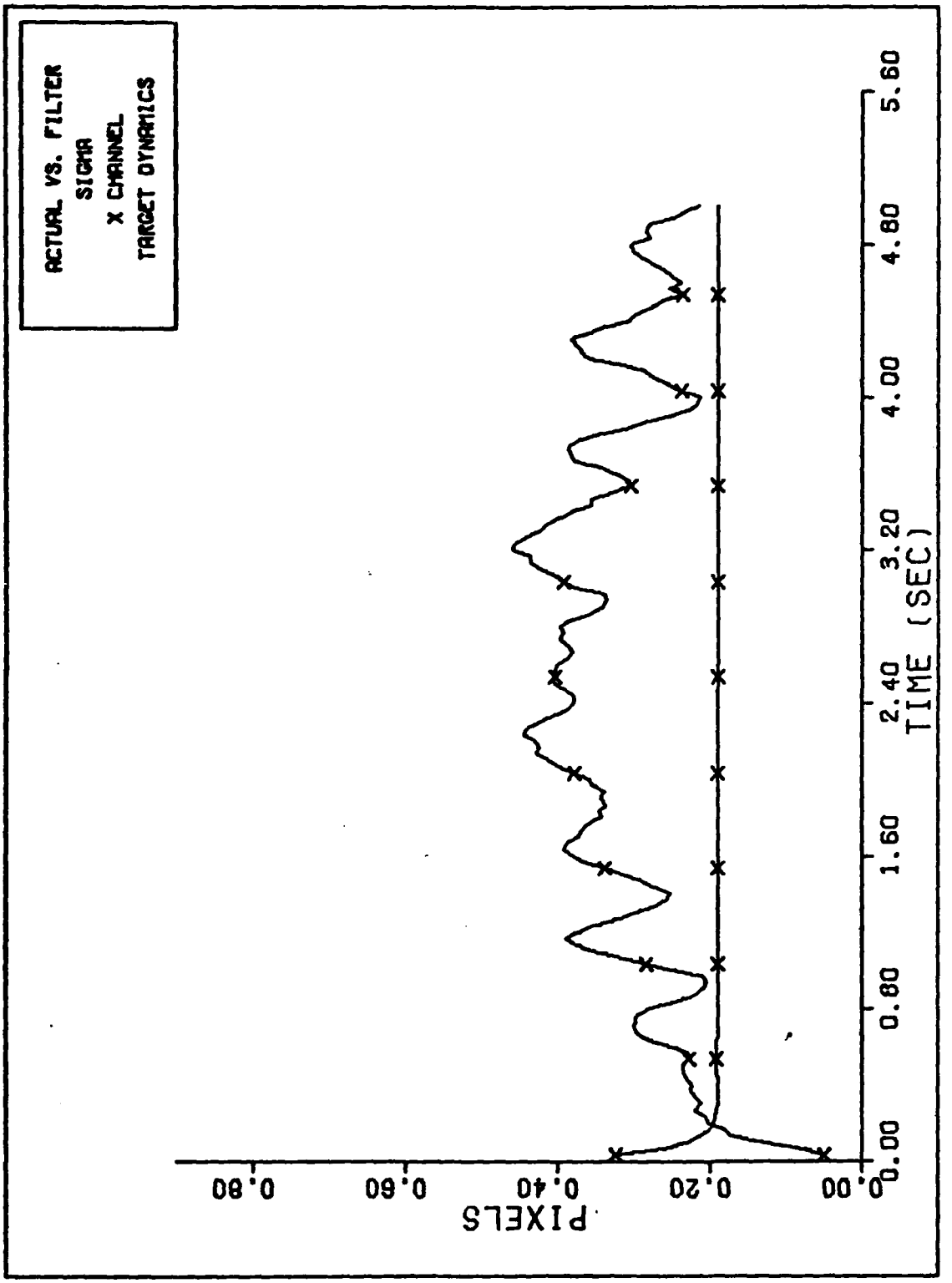
--true and filter-indicated standard deviation of the 'x' and 'y' target velocity. These plots were used in tuning the filters to the various trajectories simulated.

Plots indicating the convergence of the variance over 20 Monte Carlo runs are included in the 20 g and 10 g θ=300 plots. These plots were used in determining how many Monte Carlo simulation runs were required in order to get meaningful results. All the convergence plots for both the Brownian motion and CTR filters were very similar in appearance and information, so only a few are included as samples.



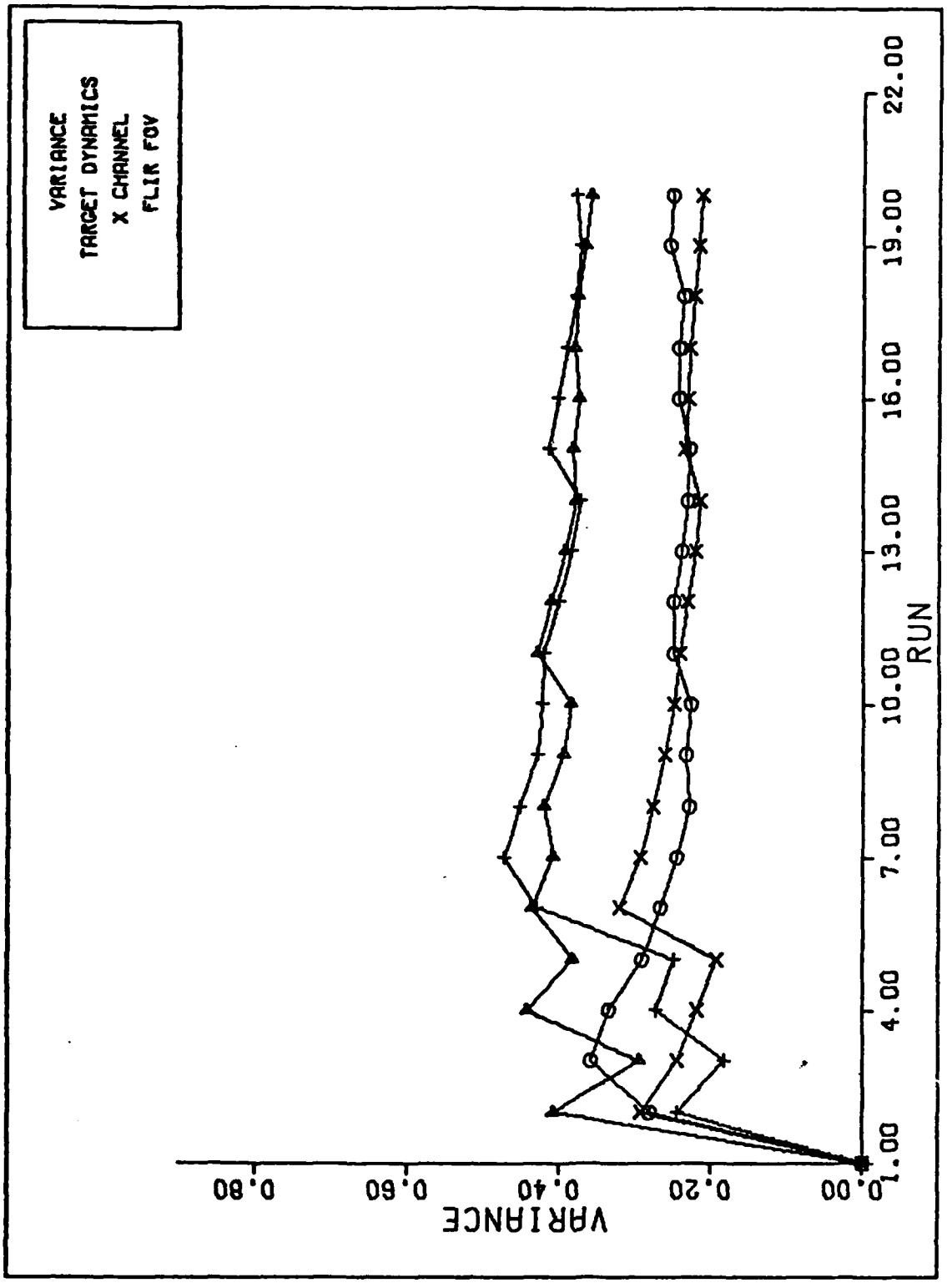
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure D-1 20 g Q=600 Performance Plot



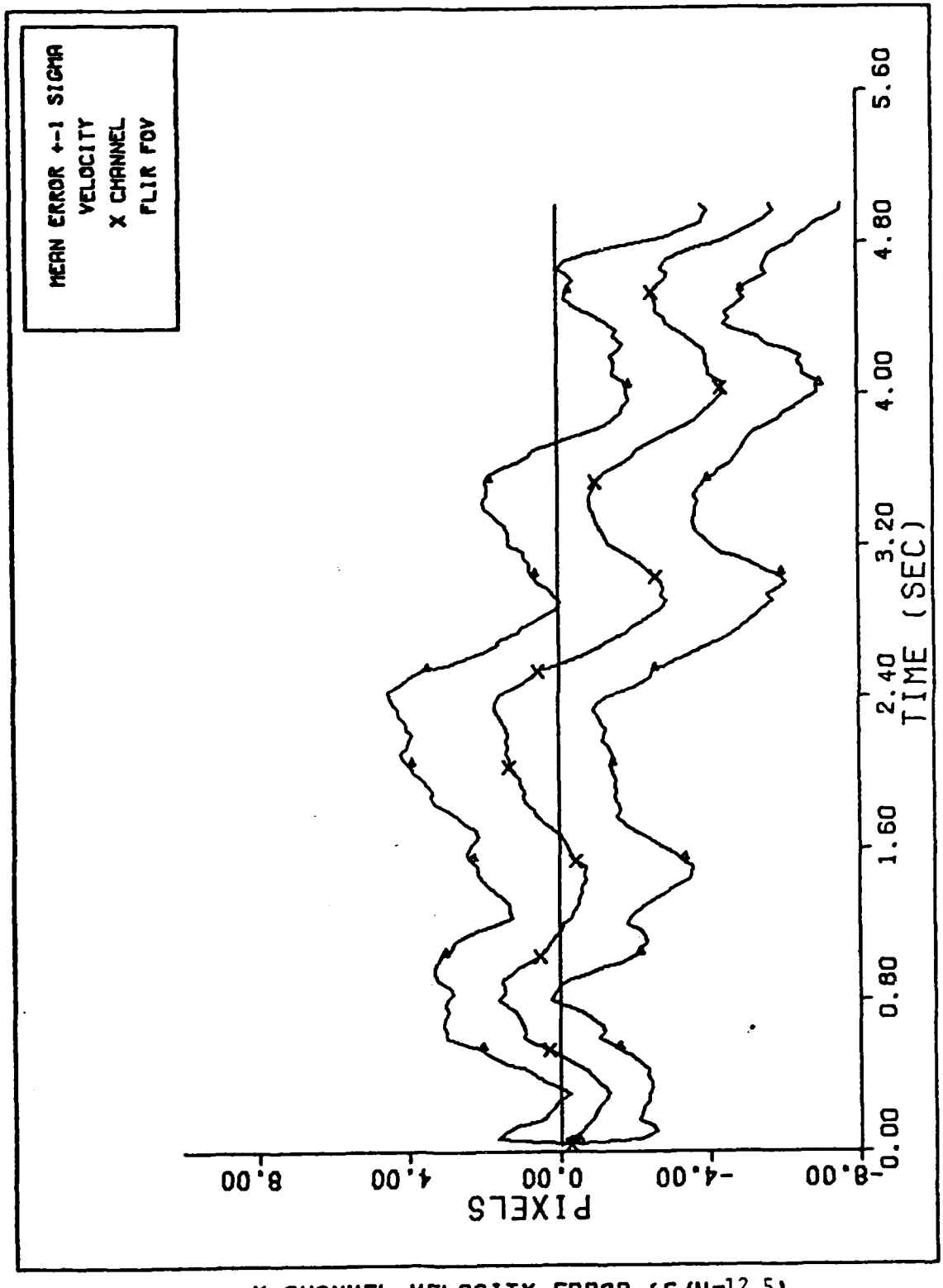
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-2 20 g Q=600 Performance Plot



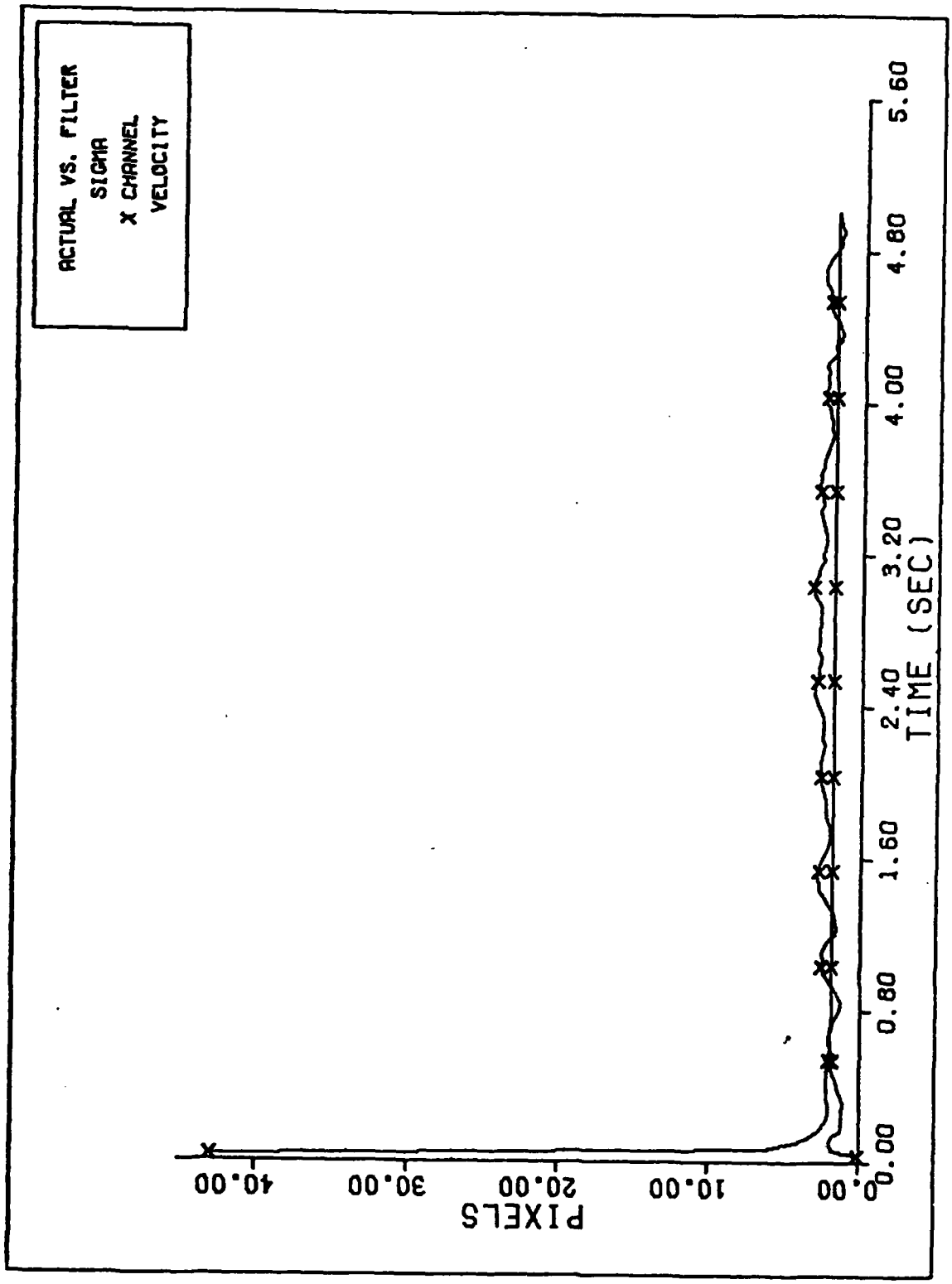
VARIANCE CONVERGENCE

Figure D-3 20 g Q=600 Performance Plot



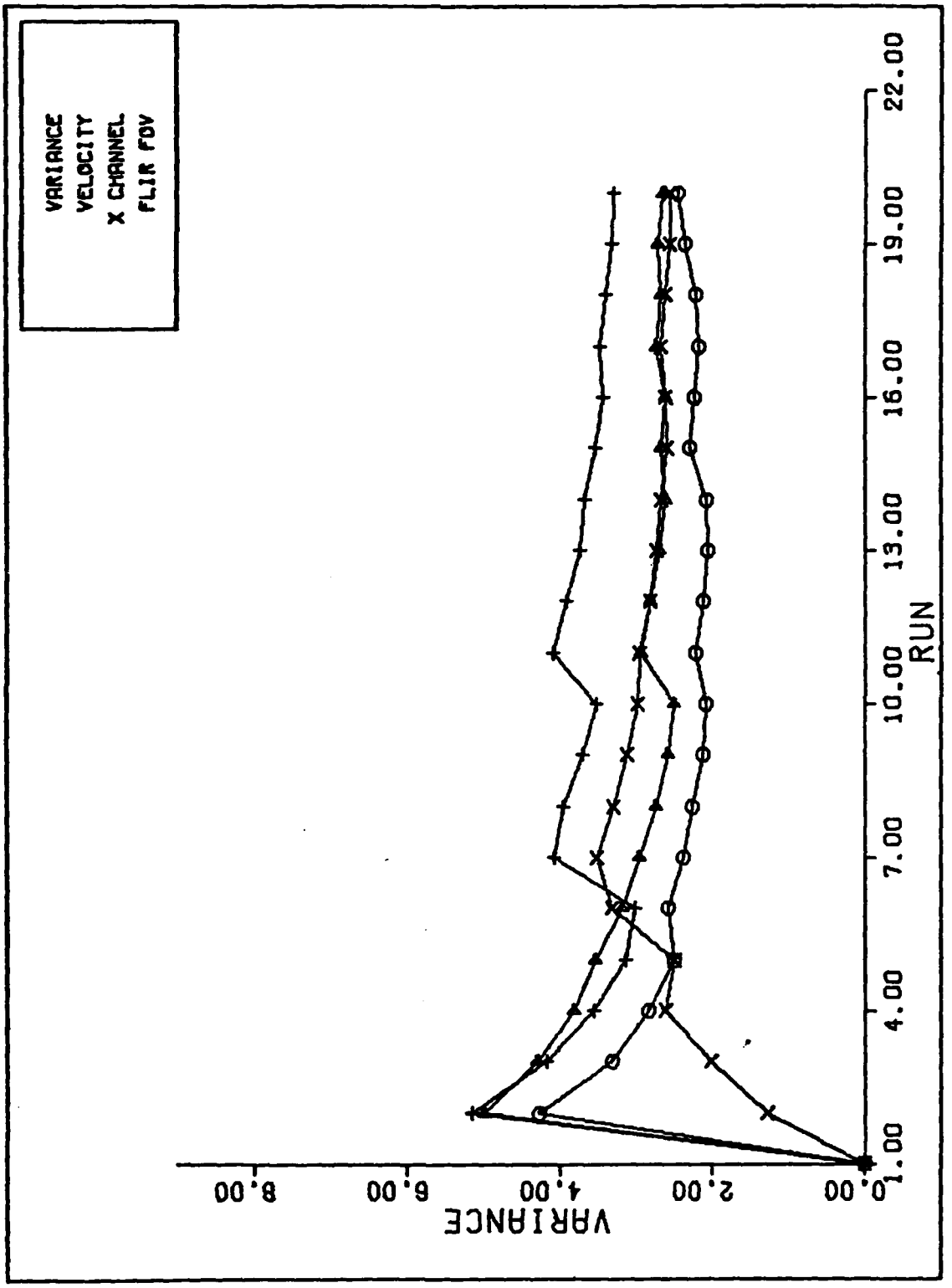
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure D-4 20 g Q=600 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-5 20 g Q=600 Performance Plot



VARIANCE CONVERGENCE

Figure D-6 20 g Q=600 Performance Plot

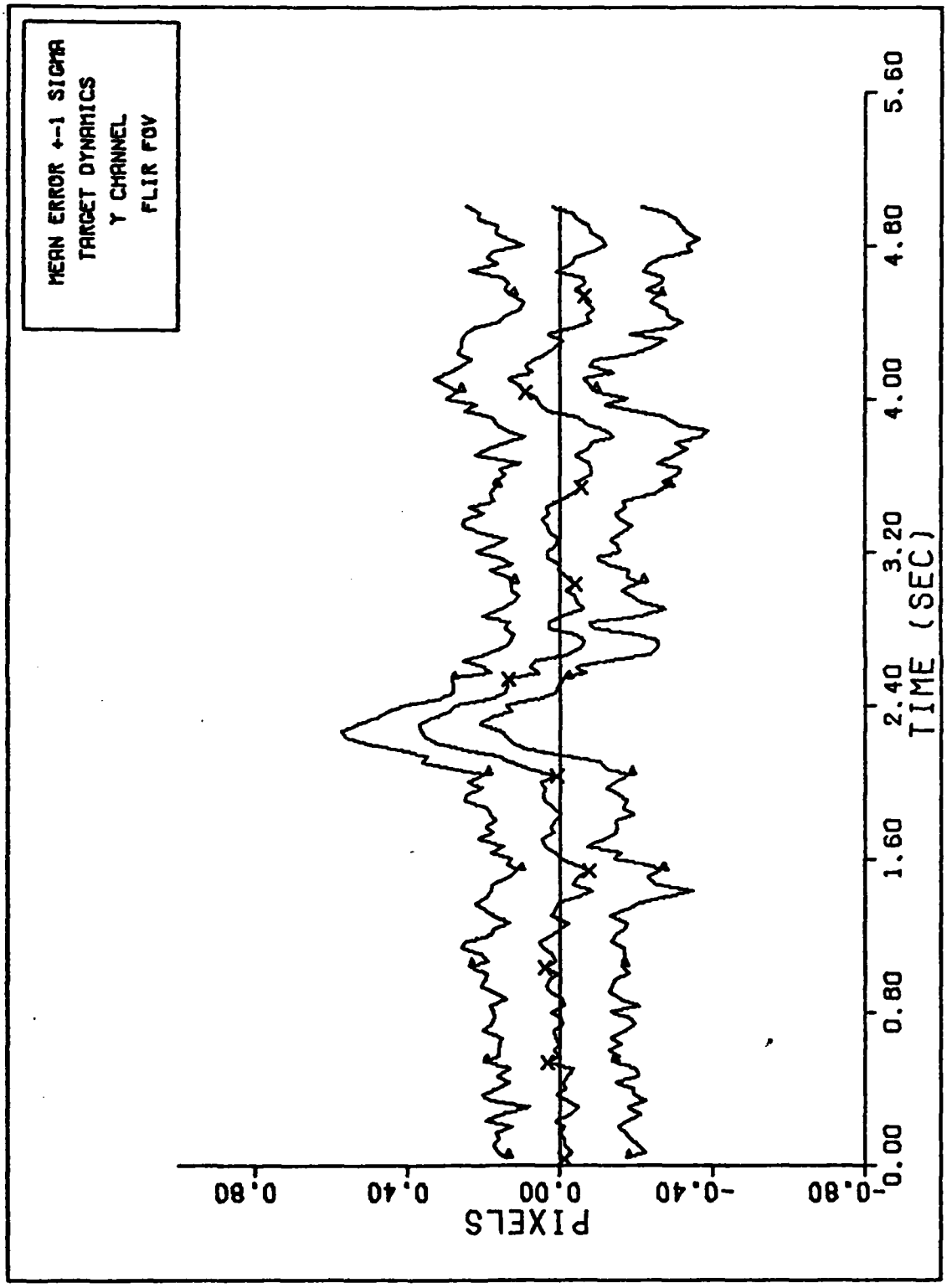
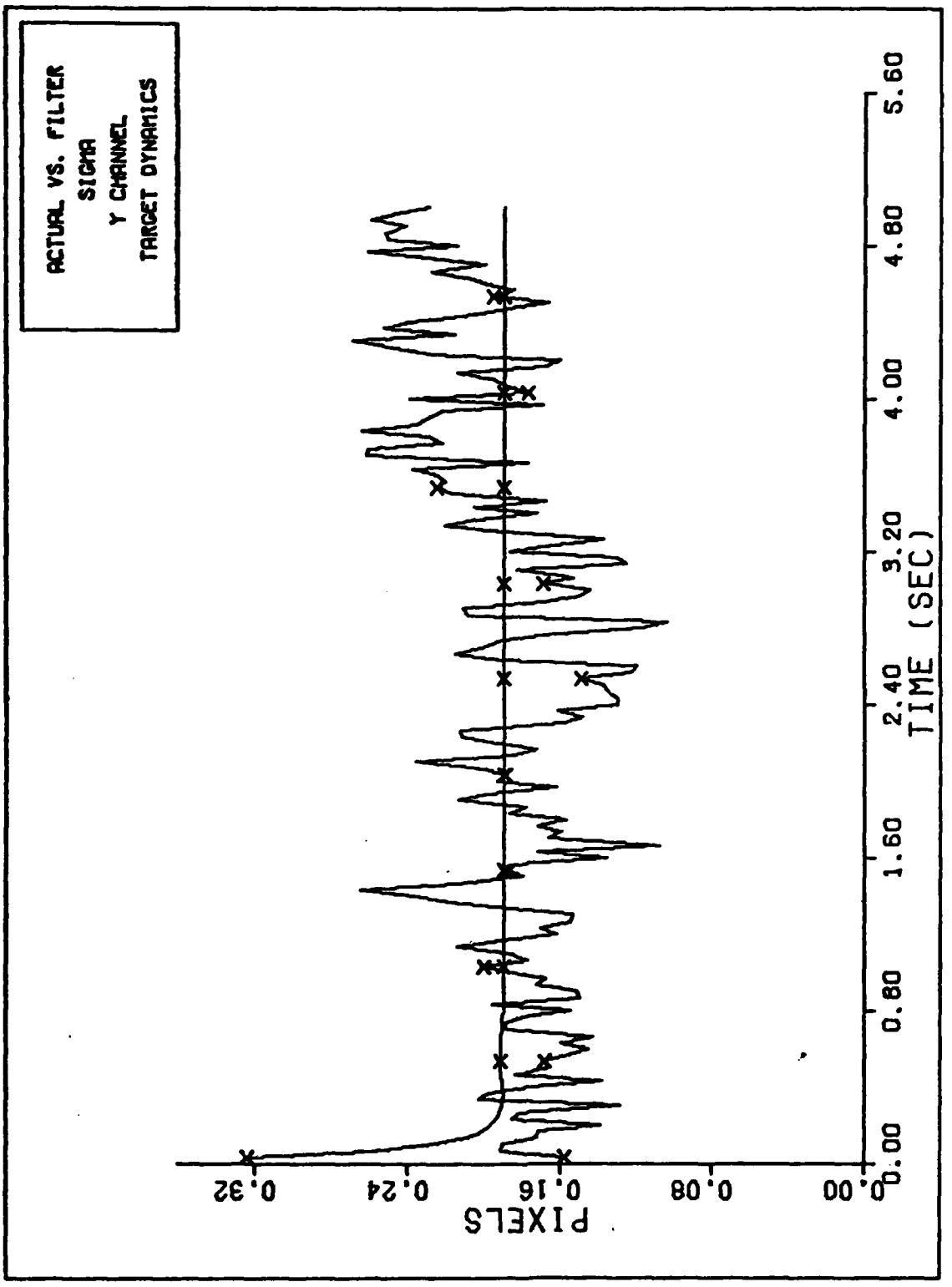
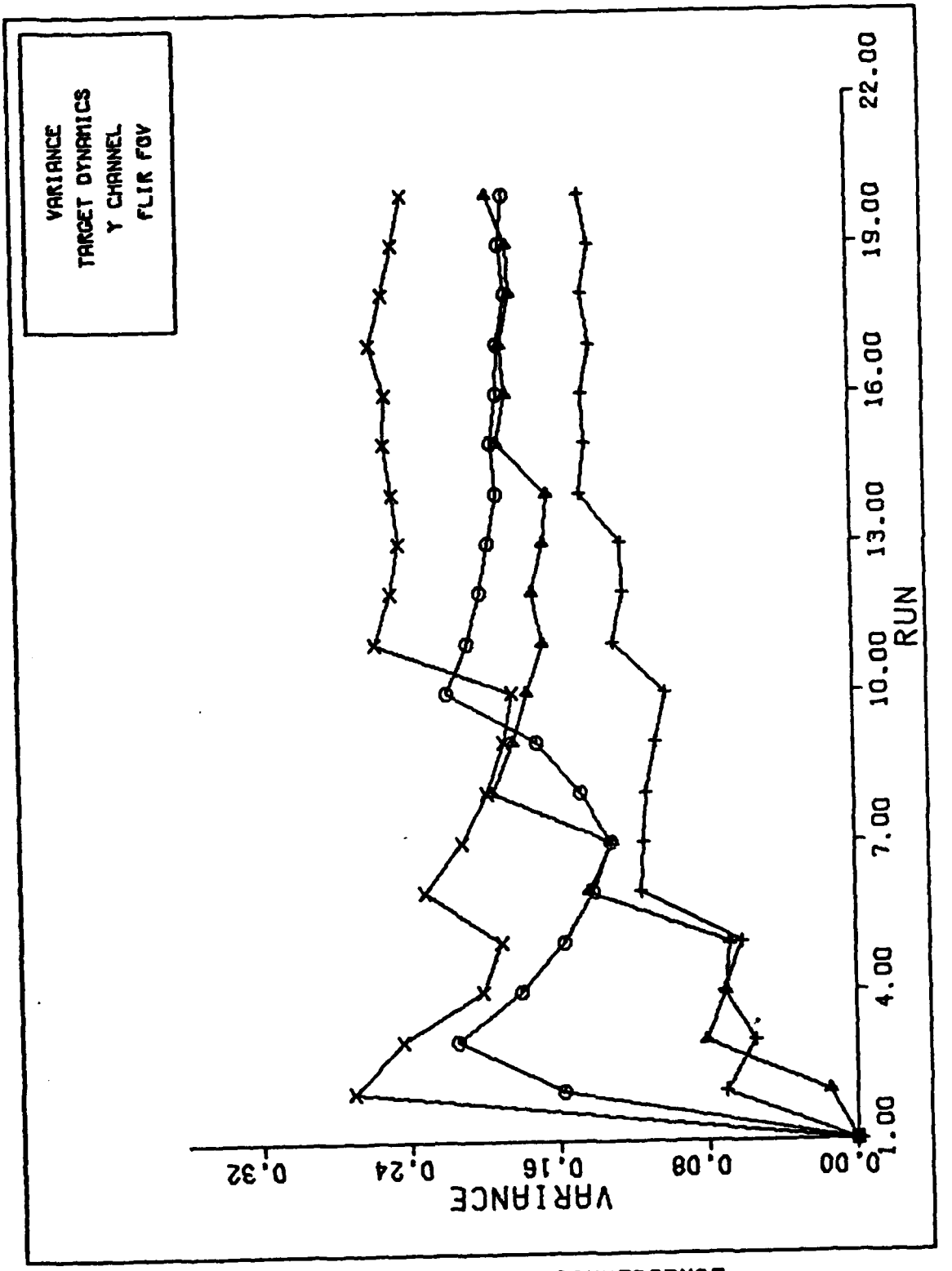


Figure D-7 20 g Q=600 Performance Plot



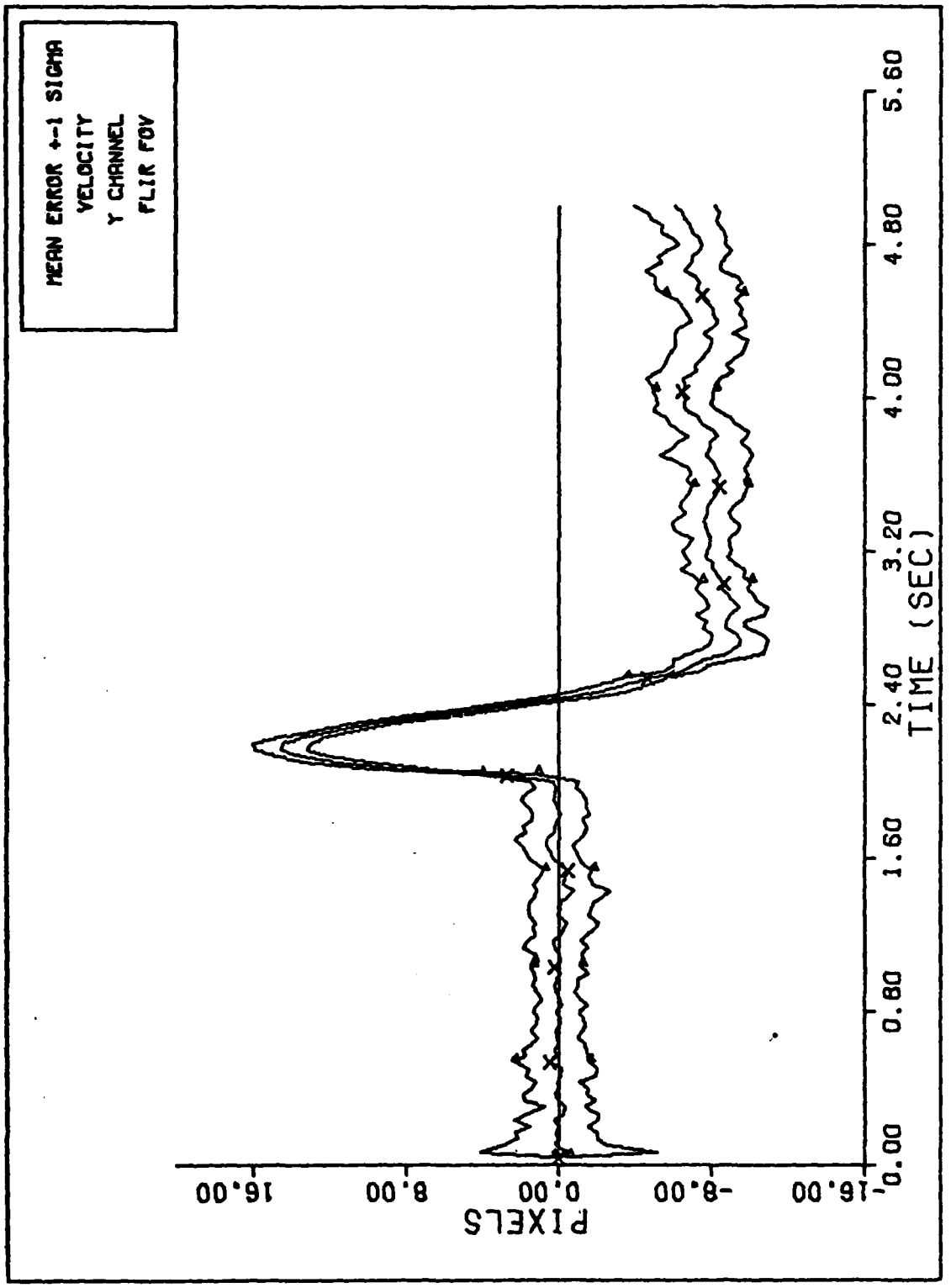
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-8 20 g Q=600 Performance Plot



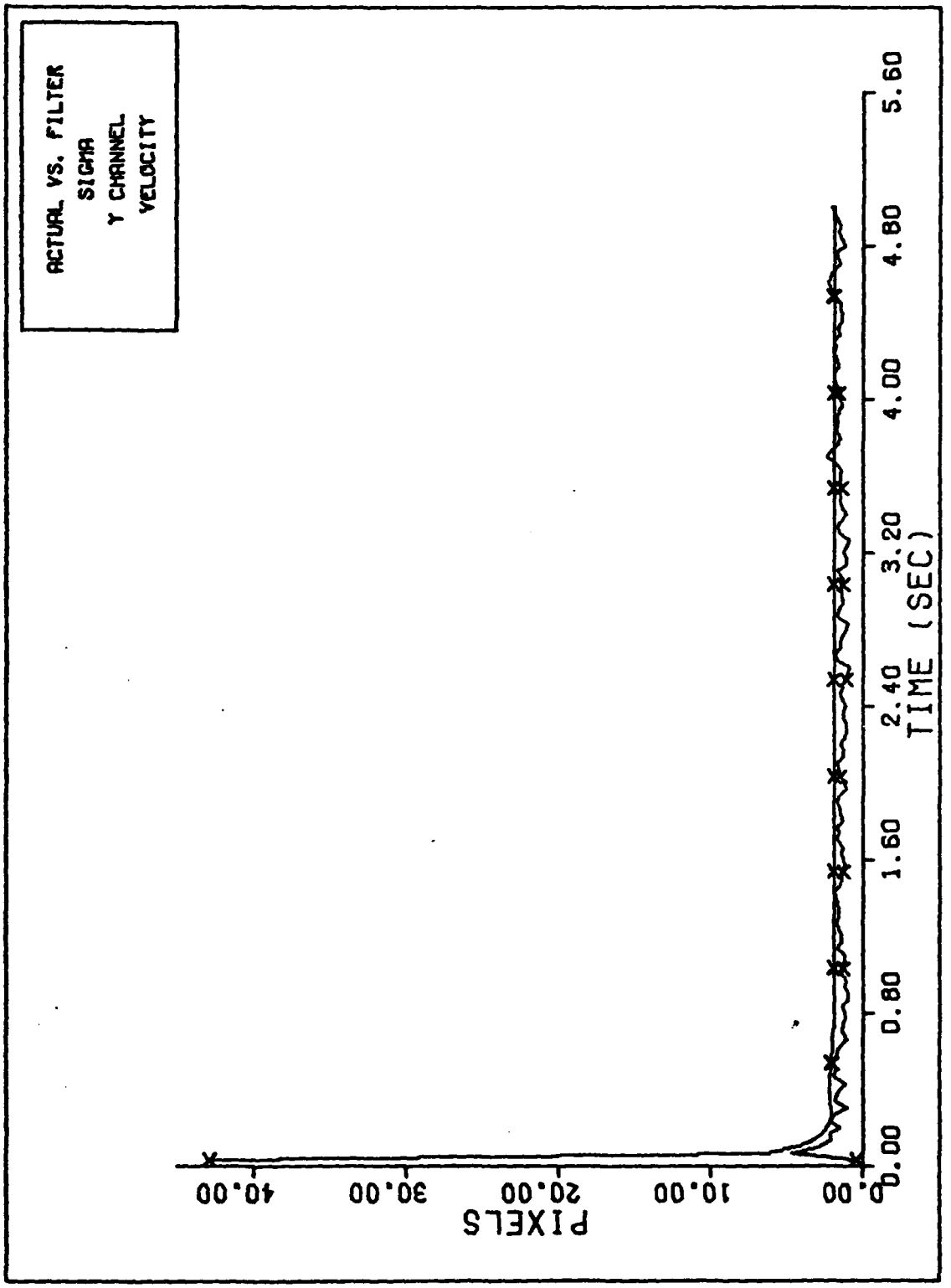
VARIANCE CONVERGENCE

Figure D-9 20 g Q=600 Performance Plot



Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure D-10 20 g Q=600 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure D-11 20 g Q=600 Performance Plot

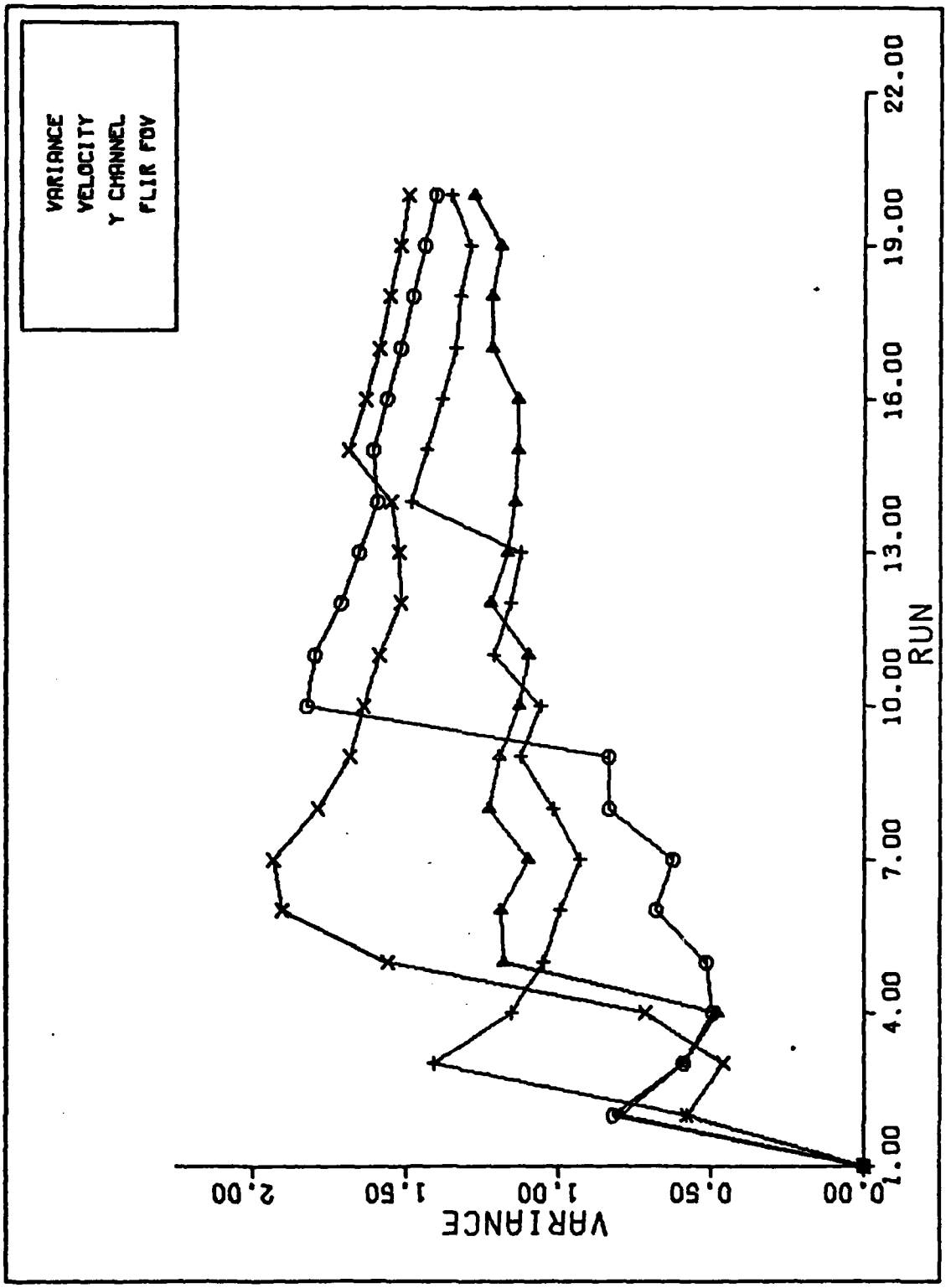
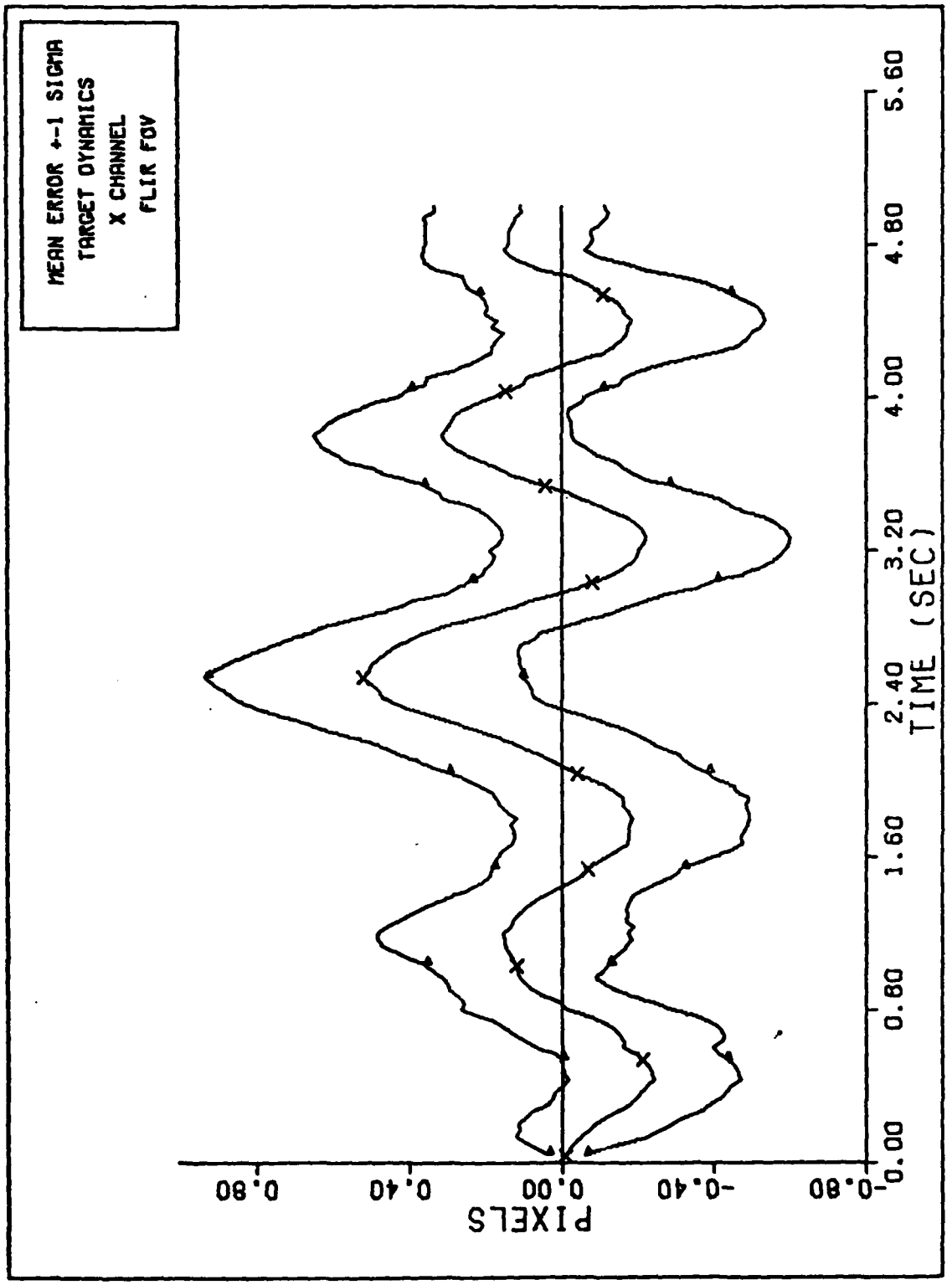
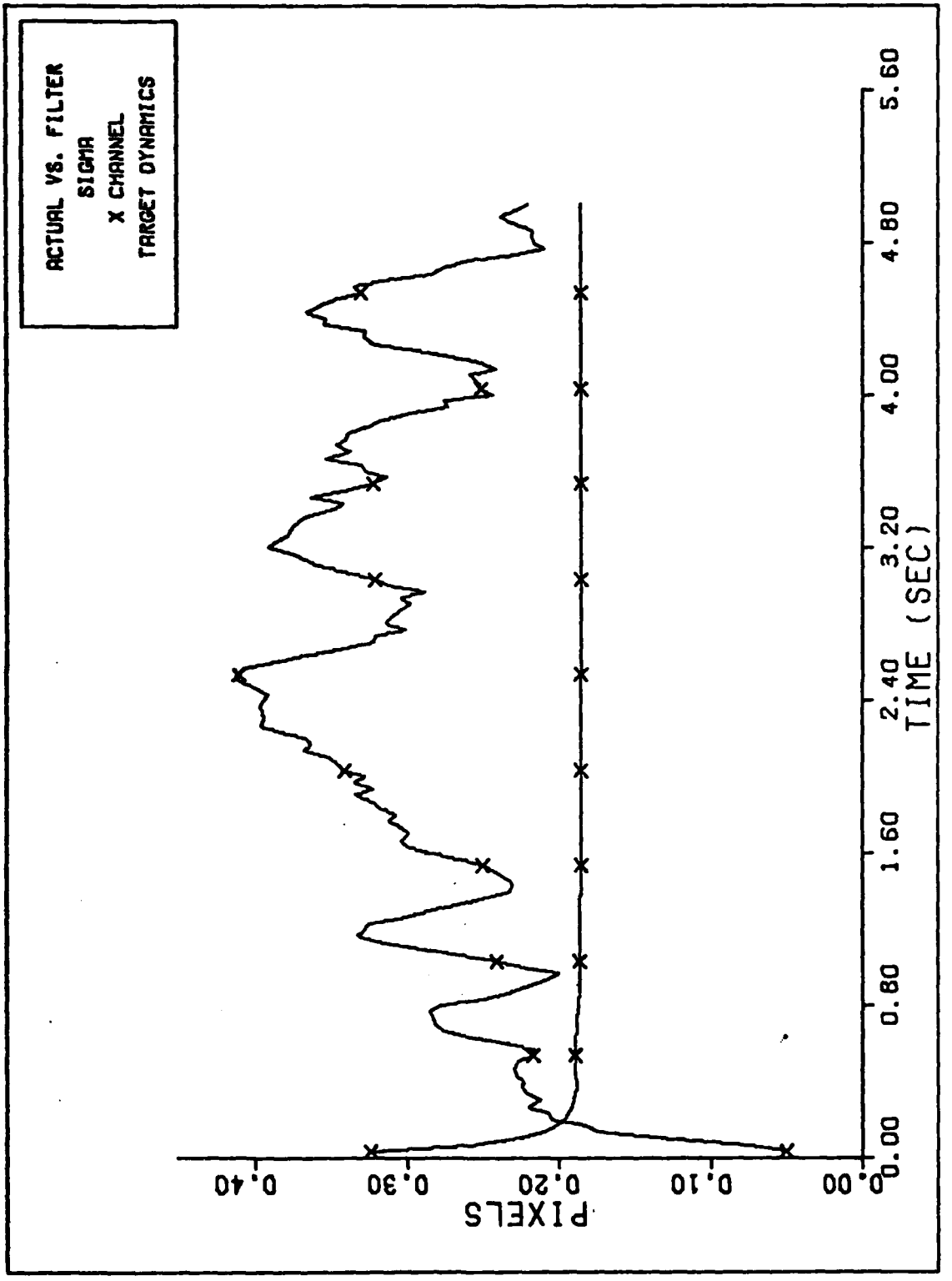


Figure D-12 20 g Q=600 Performance Plot



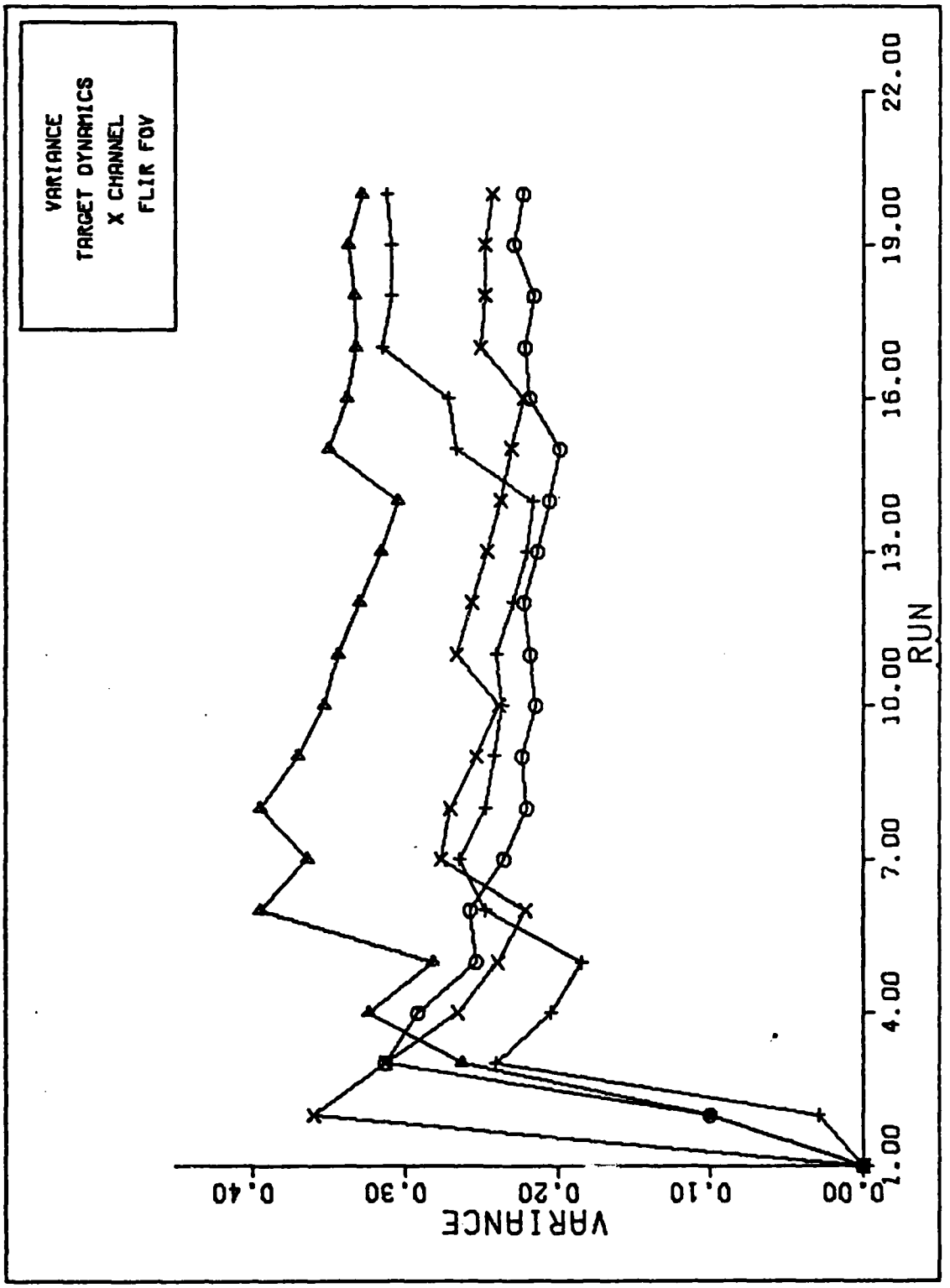
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure D-13 20 g Q=300 Performance Plot



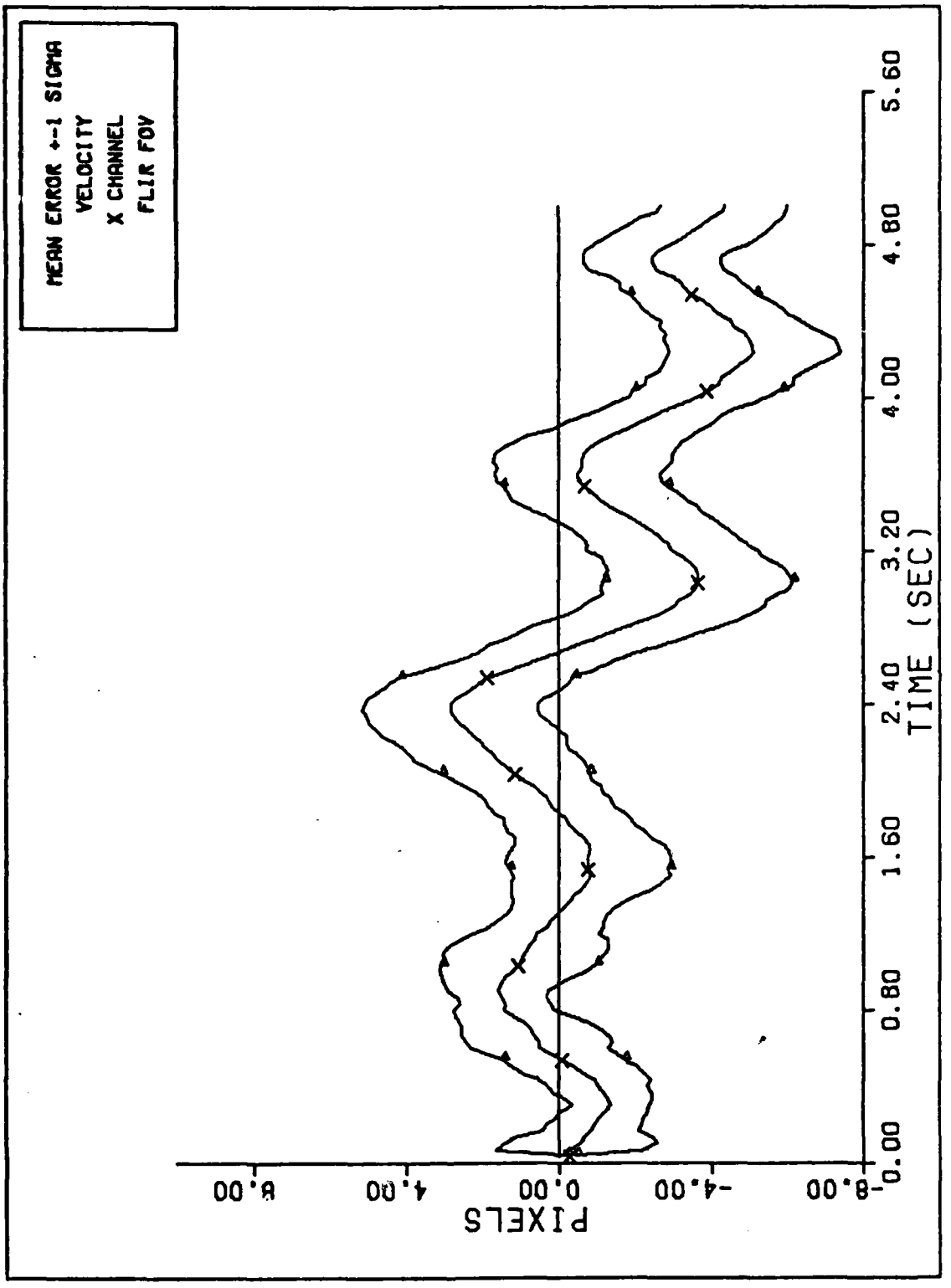
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-14 20 g Q=300 Performance Plot



VARIANCE CONVERGENCE

Figure D-15 20 g Q=300 Performance Plot



X CHANNEL VELOCITY ERROR (S/N= 12.5)

Figure D-16 20 g Q=300 Performance Plot

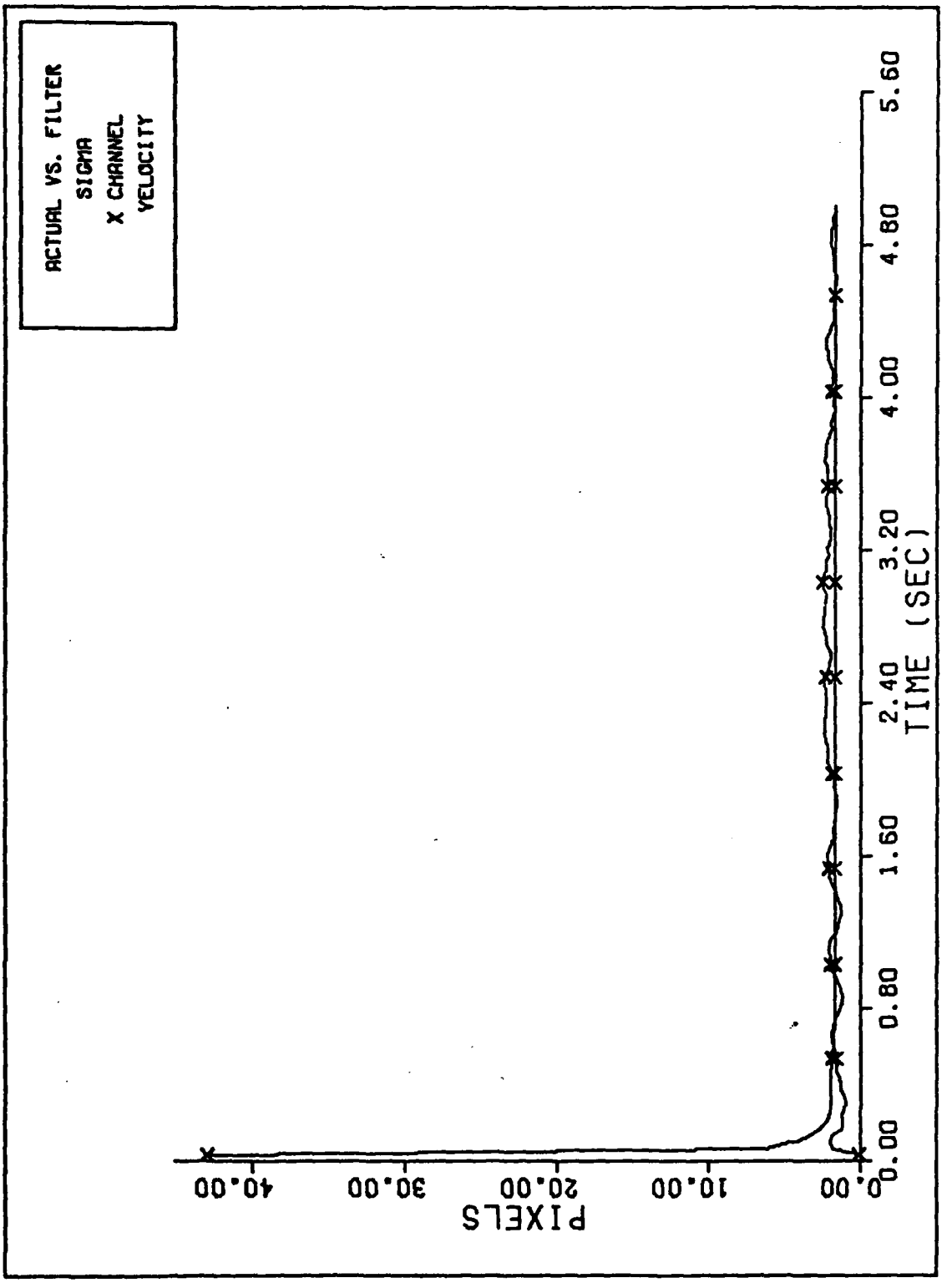


Figure D-17 20 g Q=300 Performance Plot

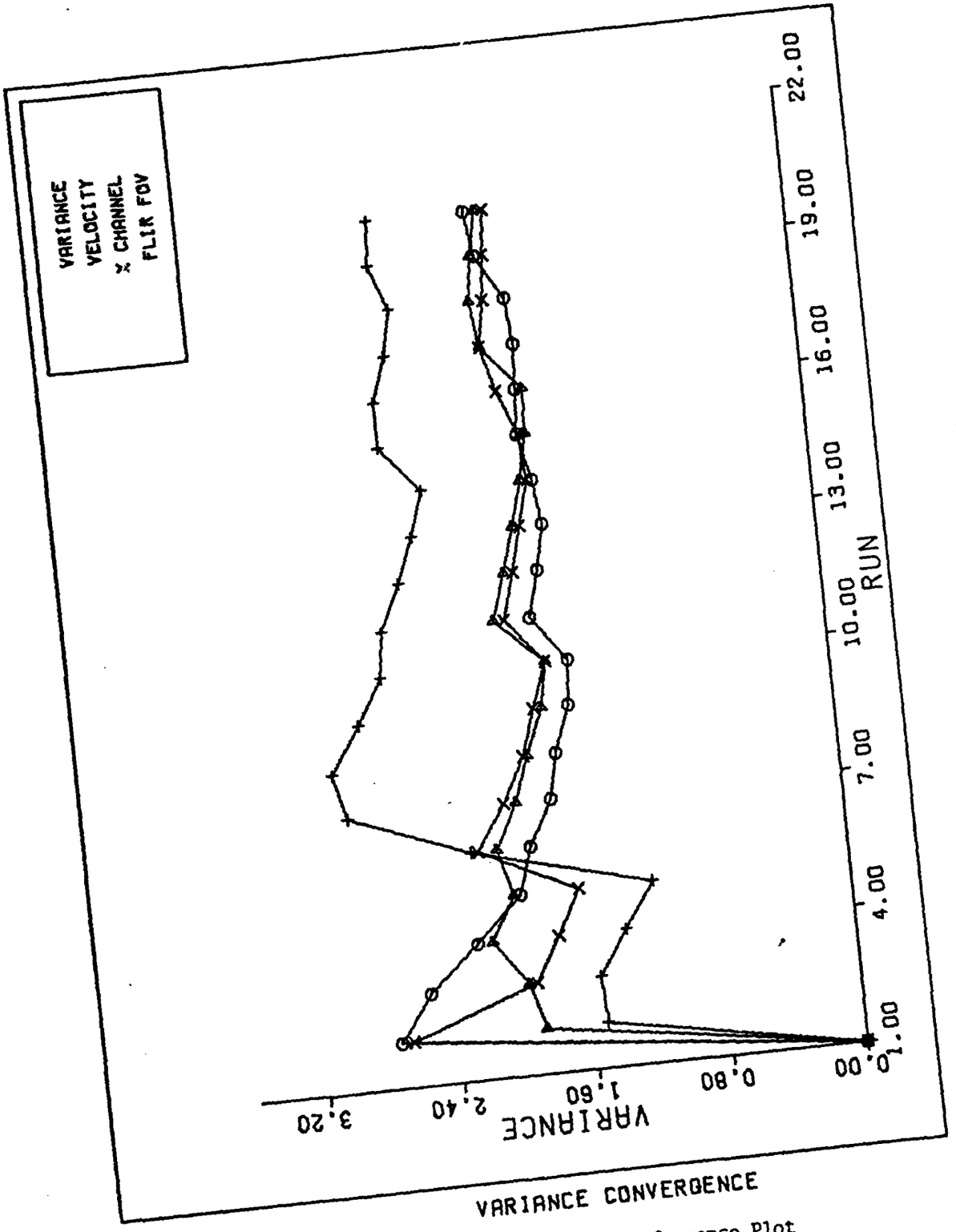
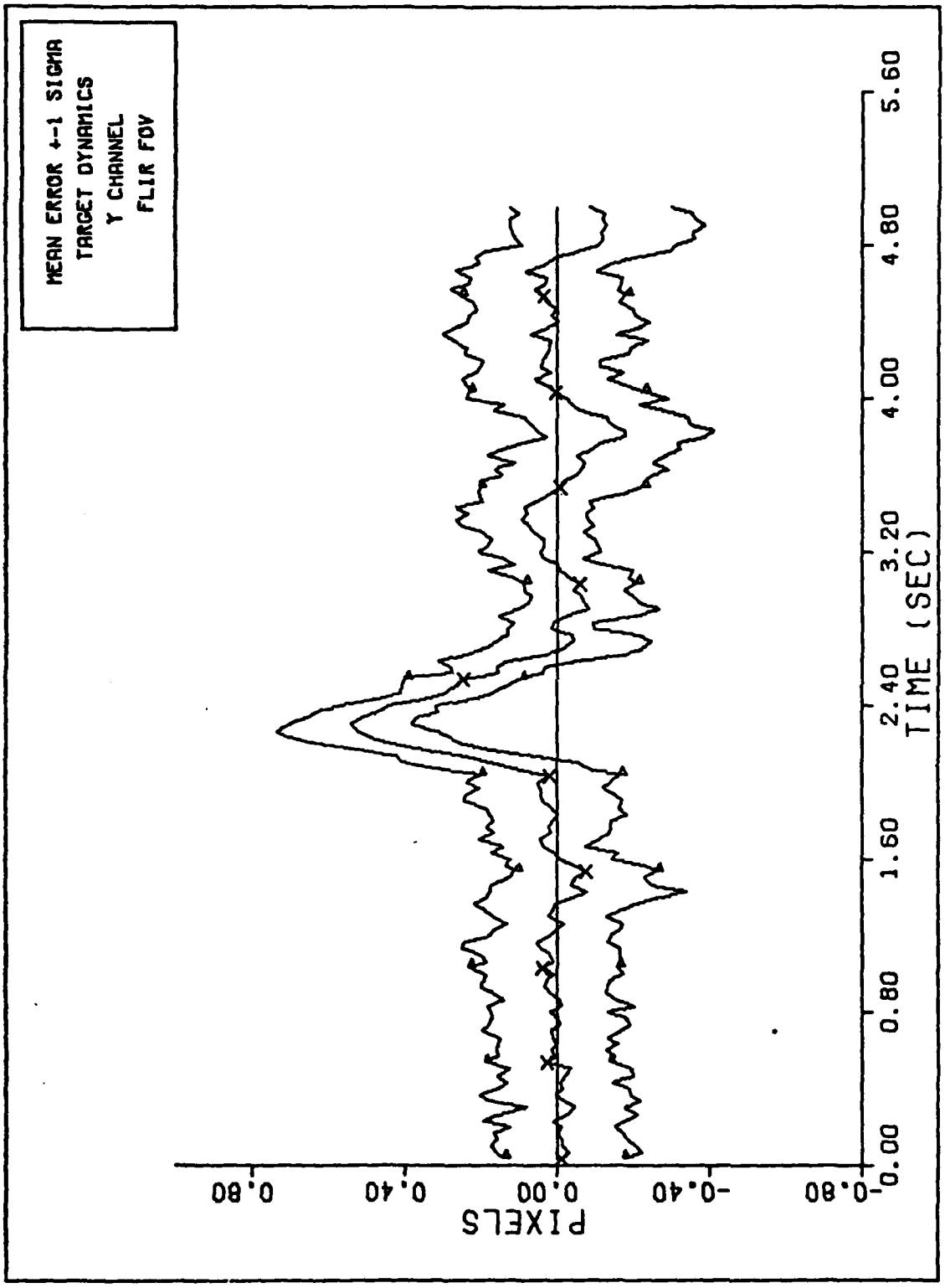


Figure D-18 20 g Q=300 Performance Plot



Y CHANNEL DYNAMICS ERROR (S/N=125)

Figure D-19 20 g Q=300 Performance Plot

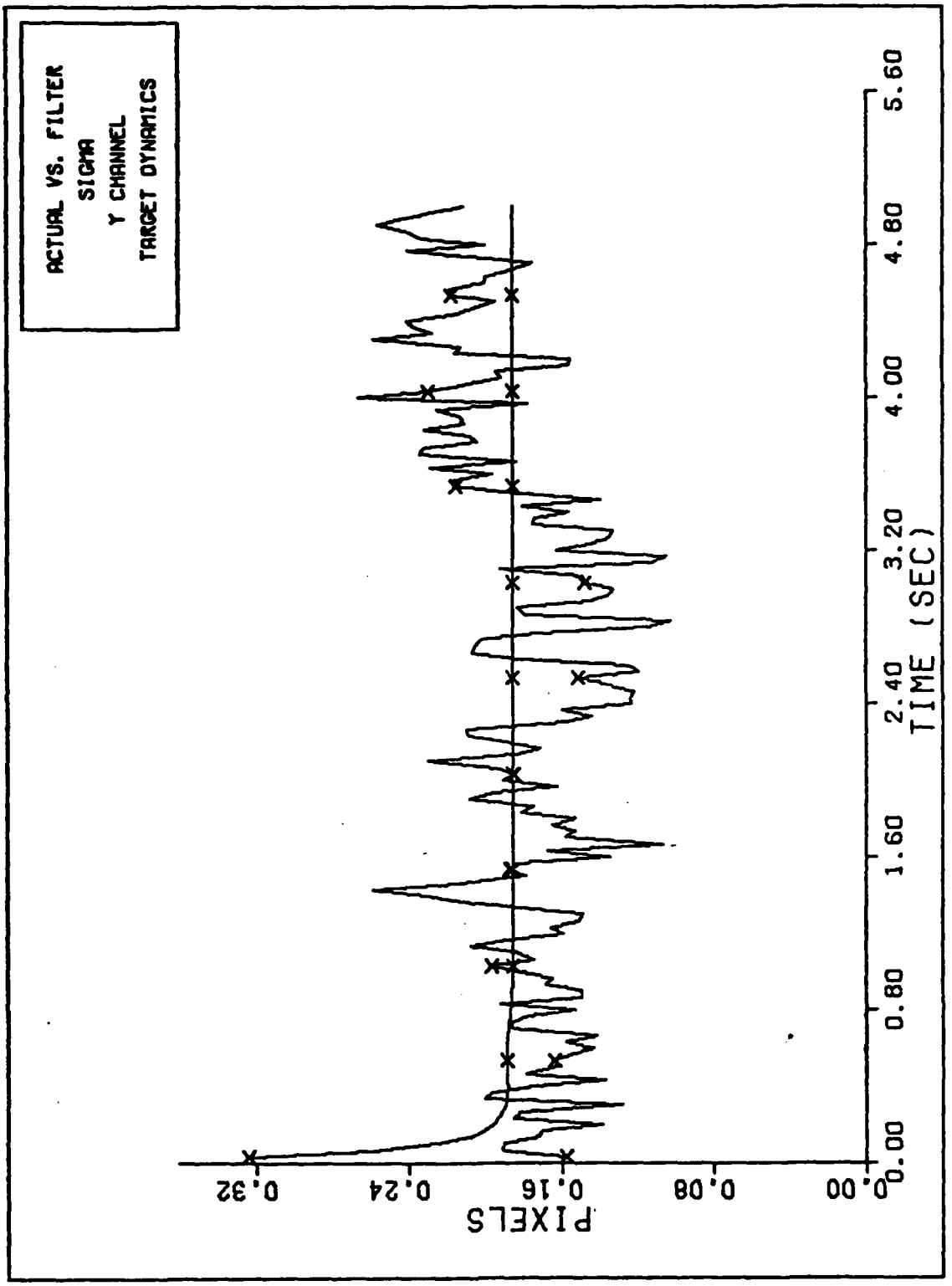
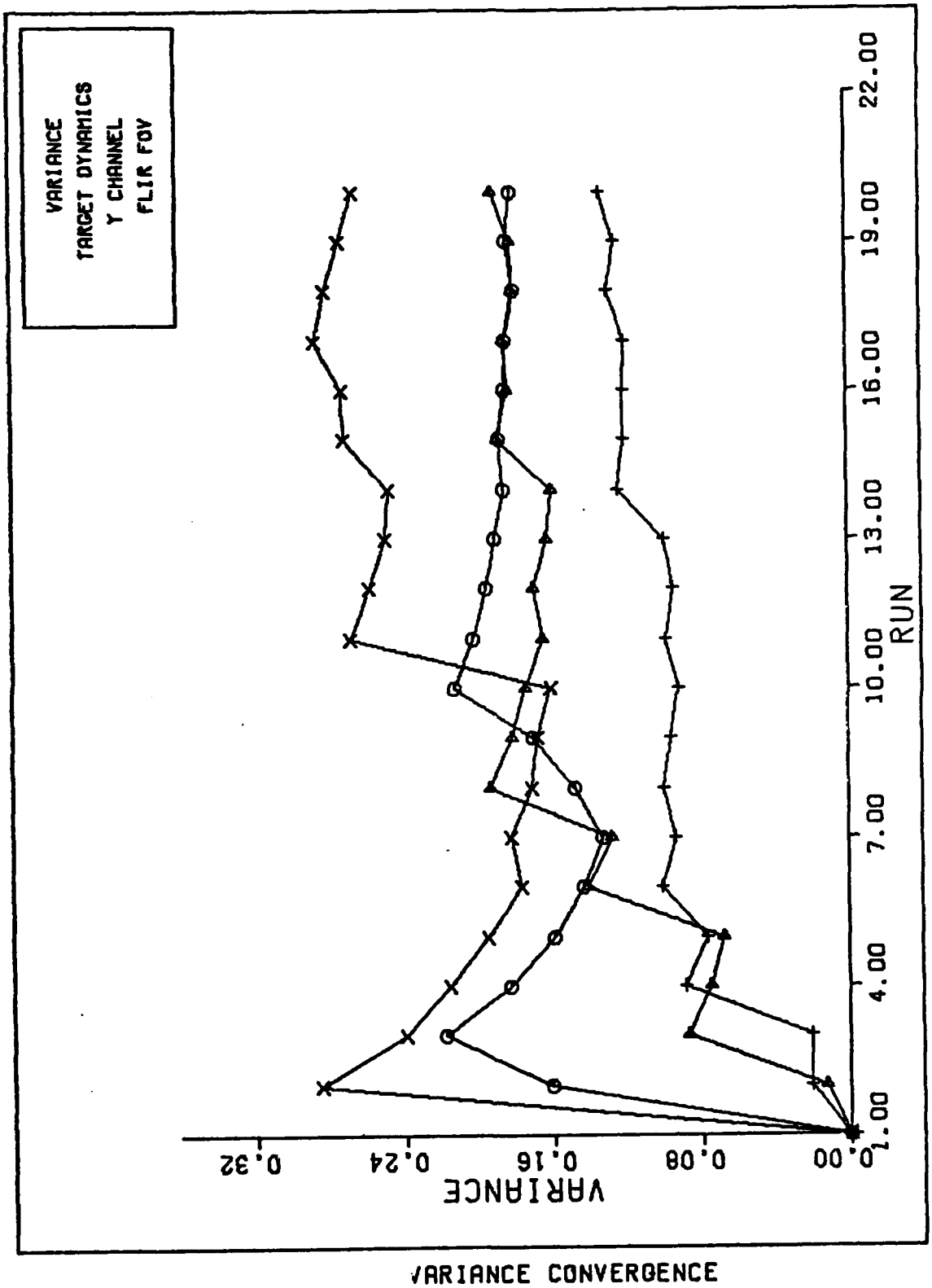


Figure D-20 20 g Q=300 Performance Plot



VARIANCE CONVERGENCE

Figure D-21 20 g Q=300 Performance Plot

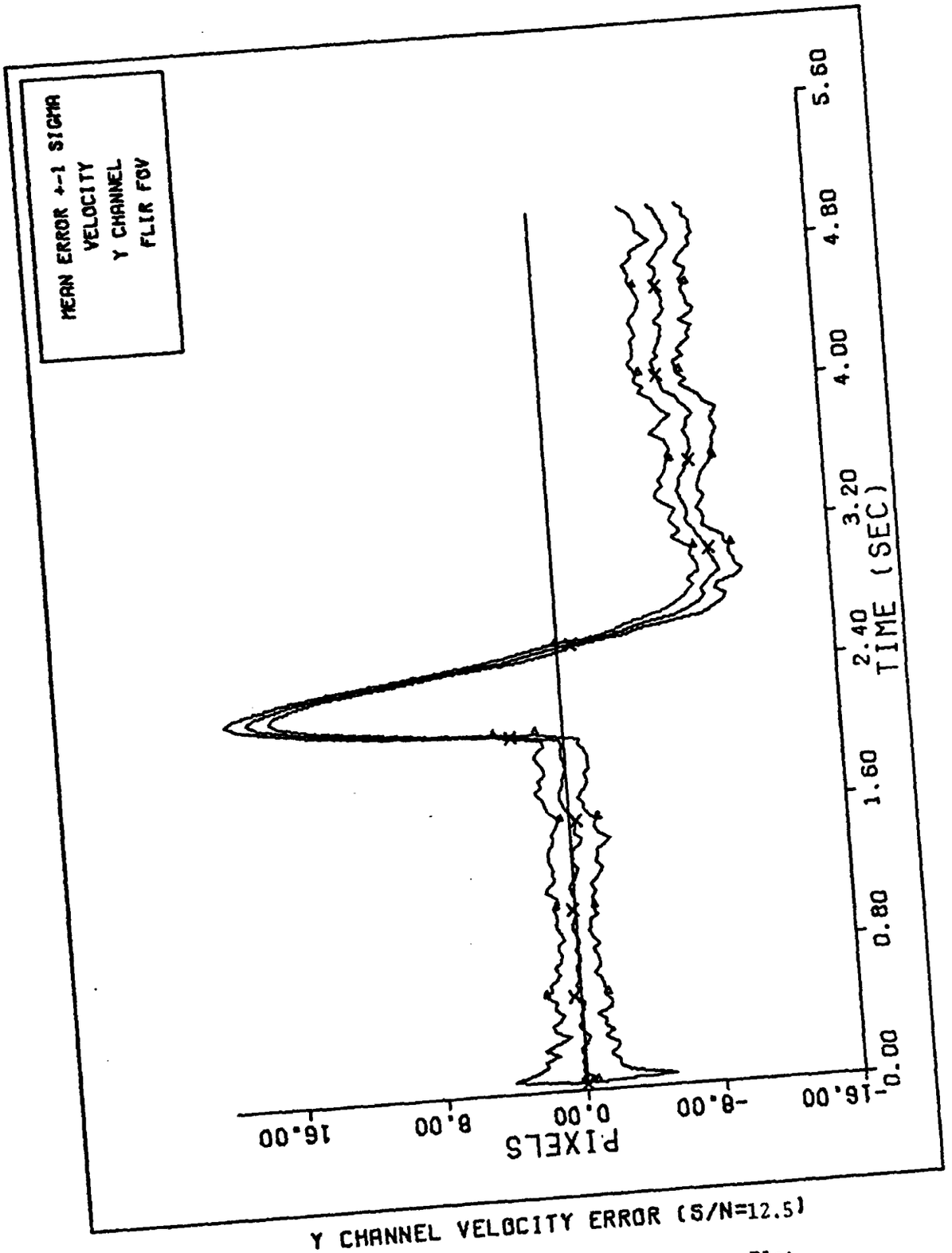
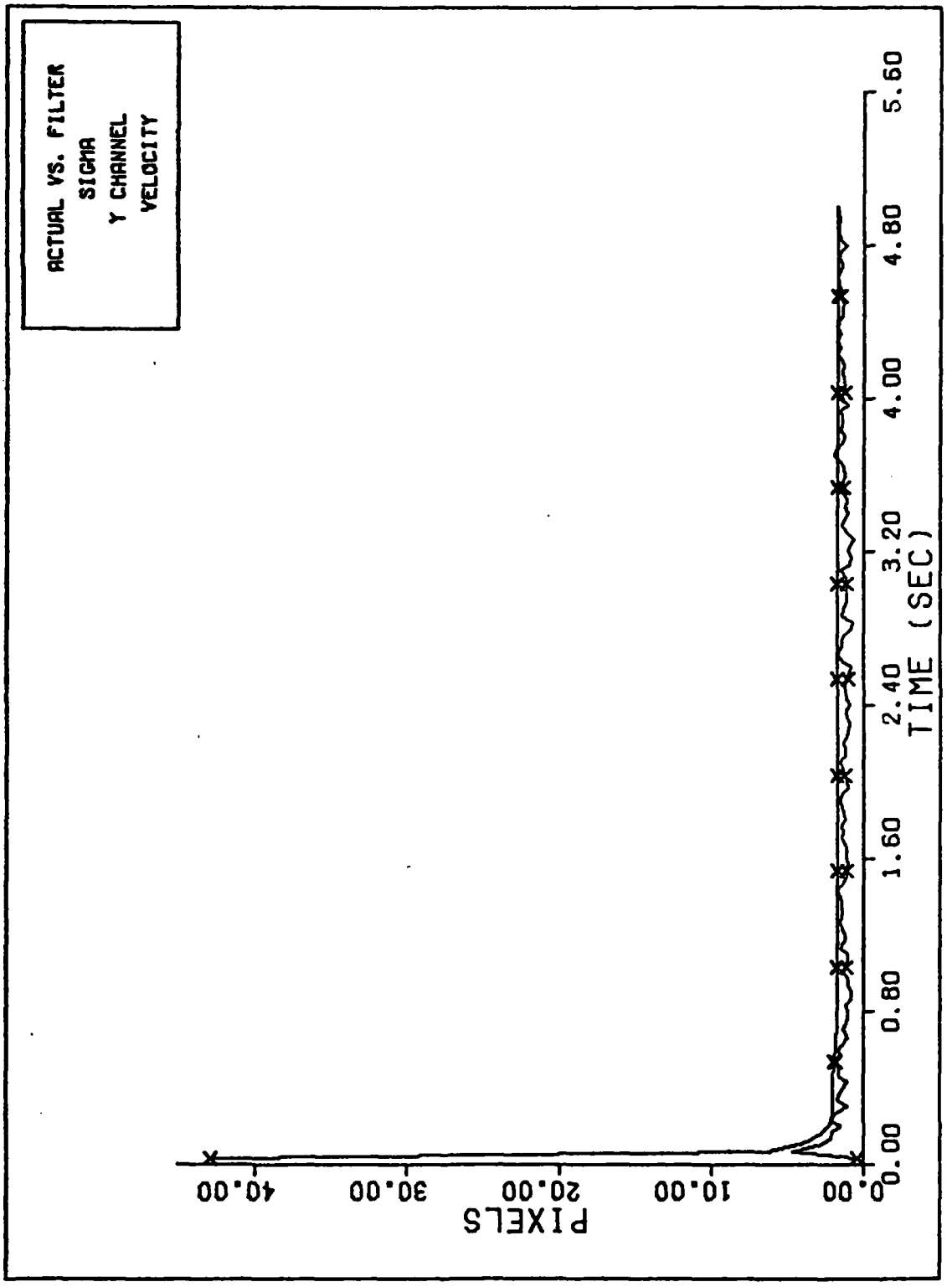


Figure D-22 20 g Q=300 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure D-23 20 g Q=300 Performance Plot

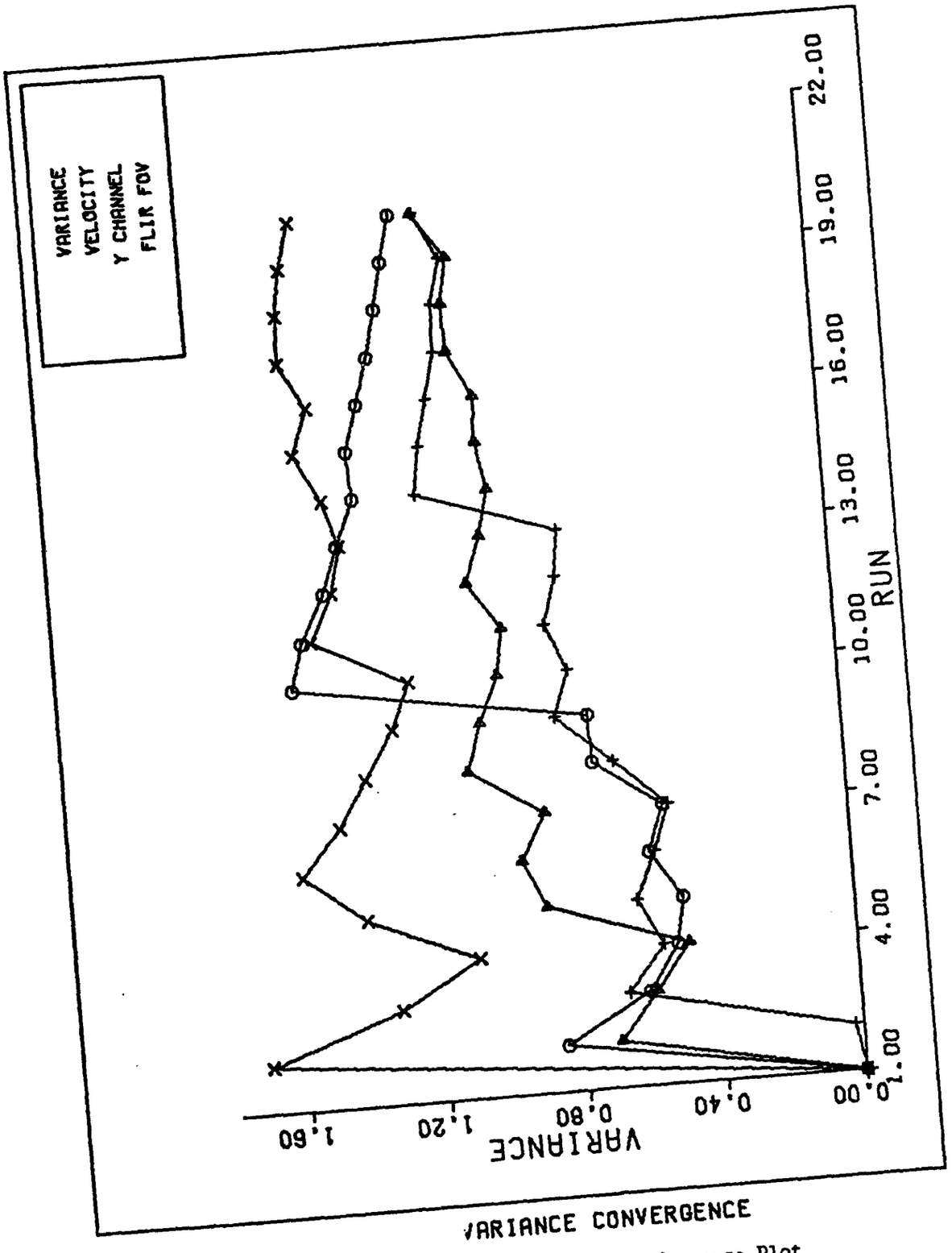


Figure D-24 20 g Q=300 Performance Plot

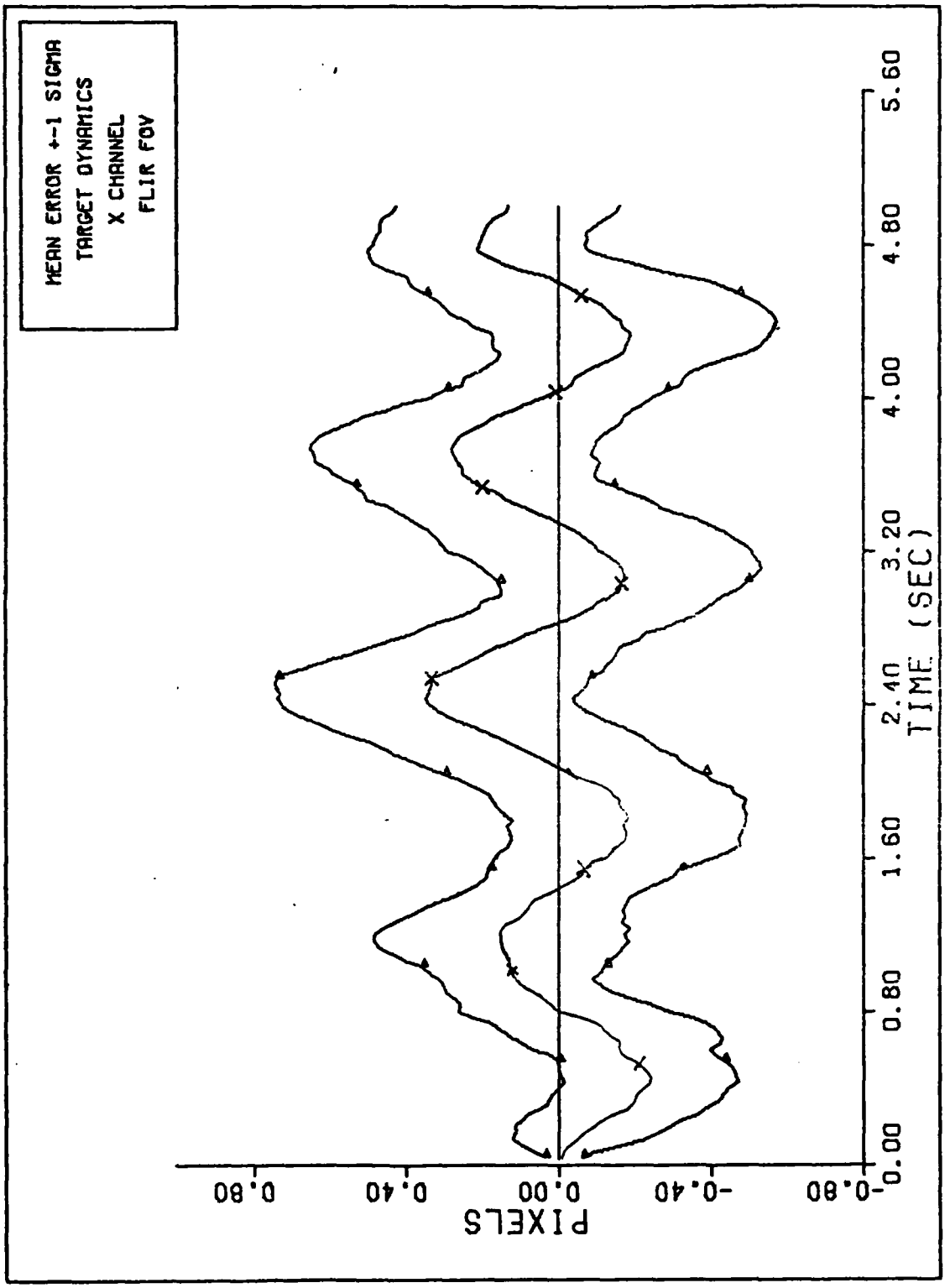
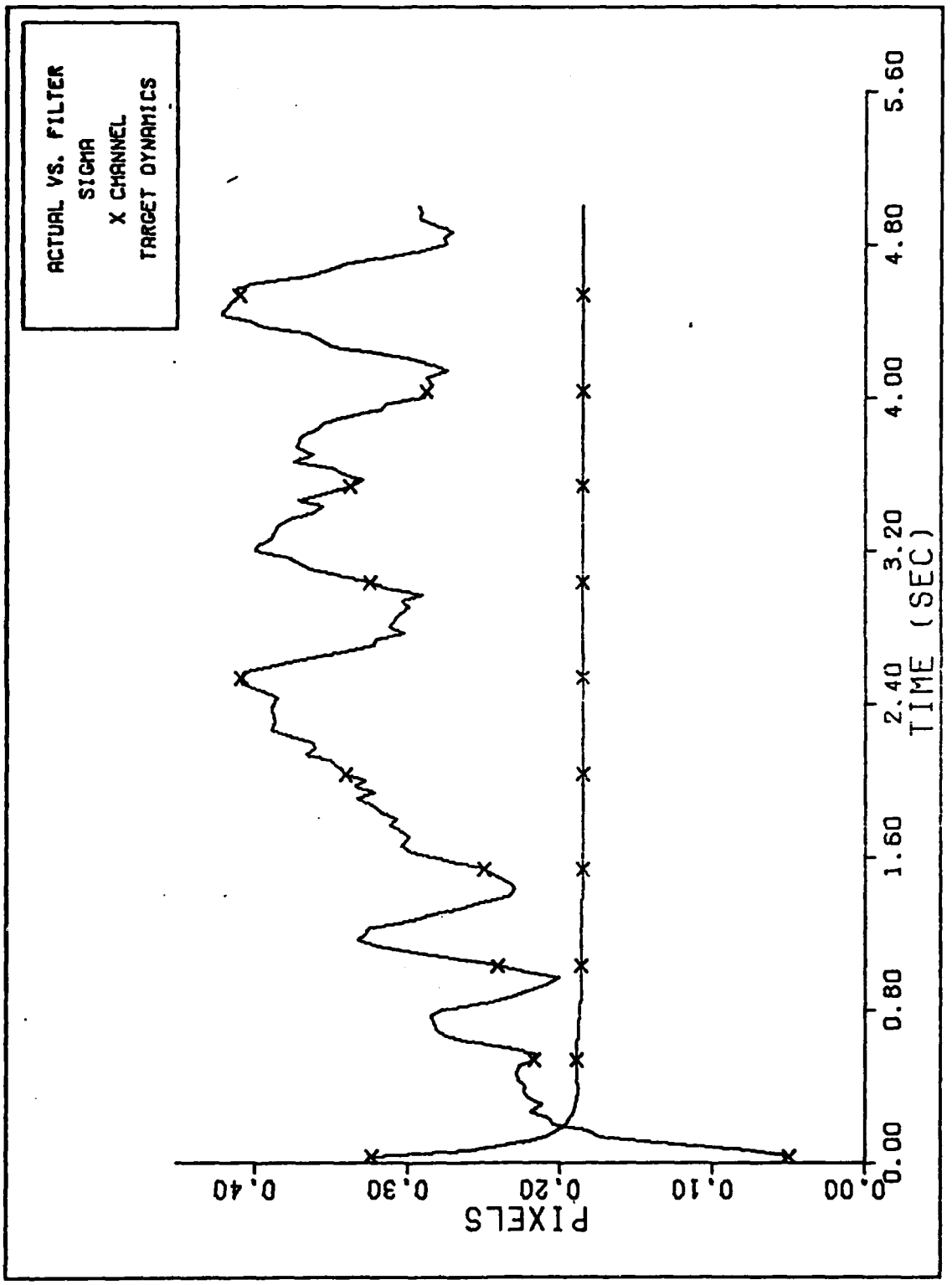
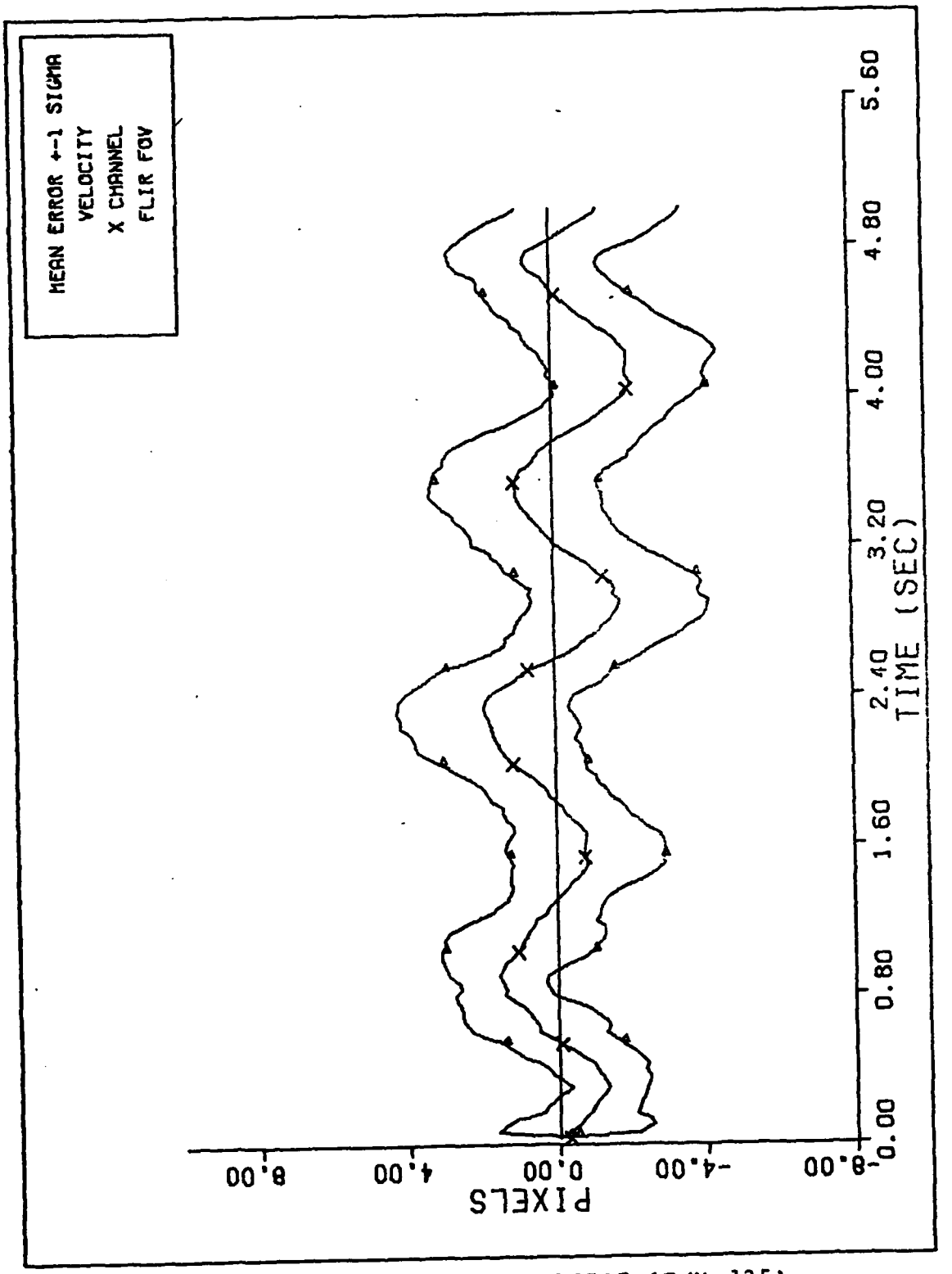


Figure D-25 10 g Q=300 Performance Plot



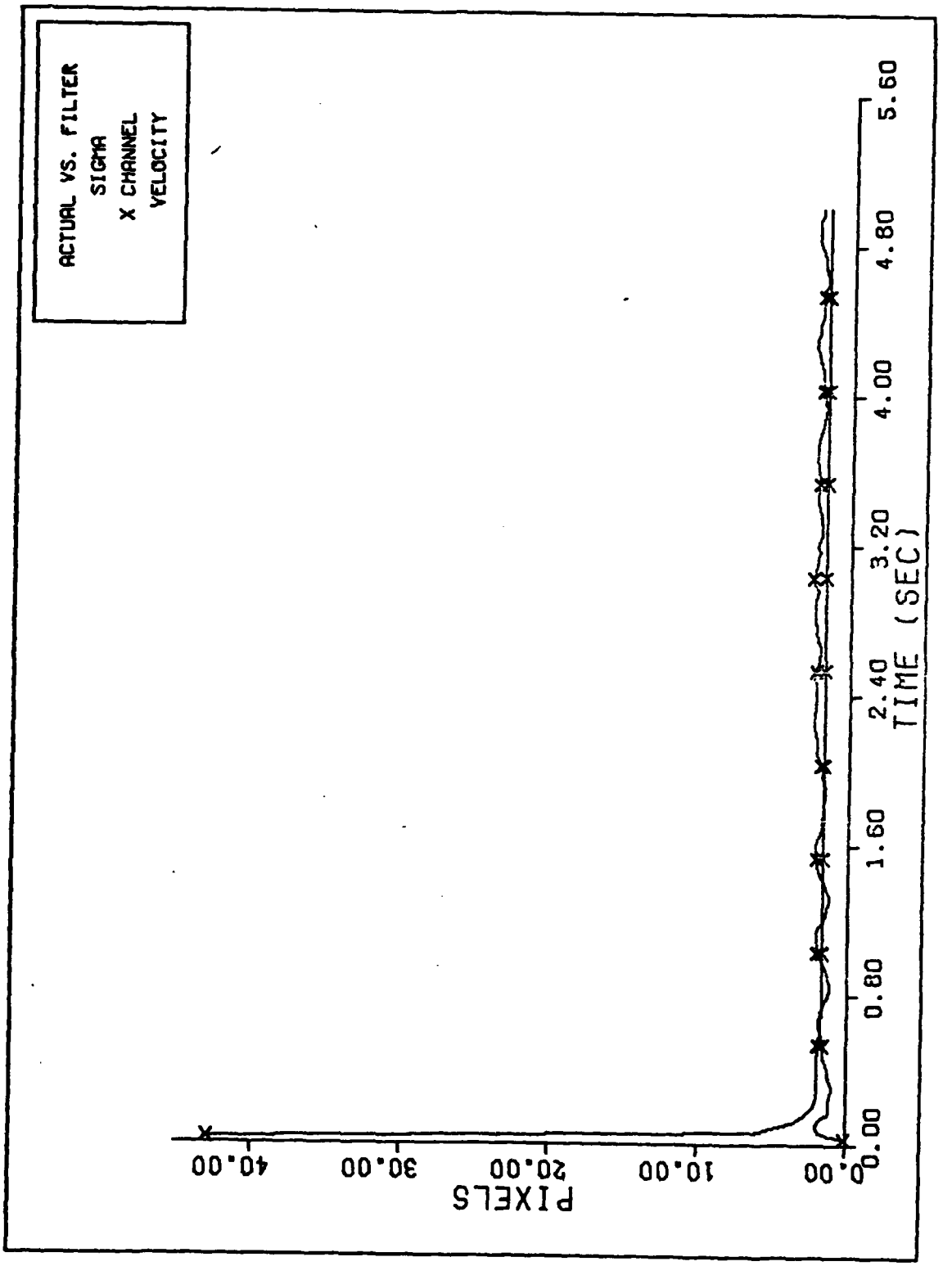
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-26 10 g Q=300 Performance Plot



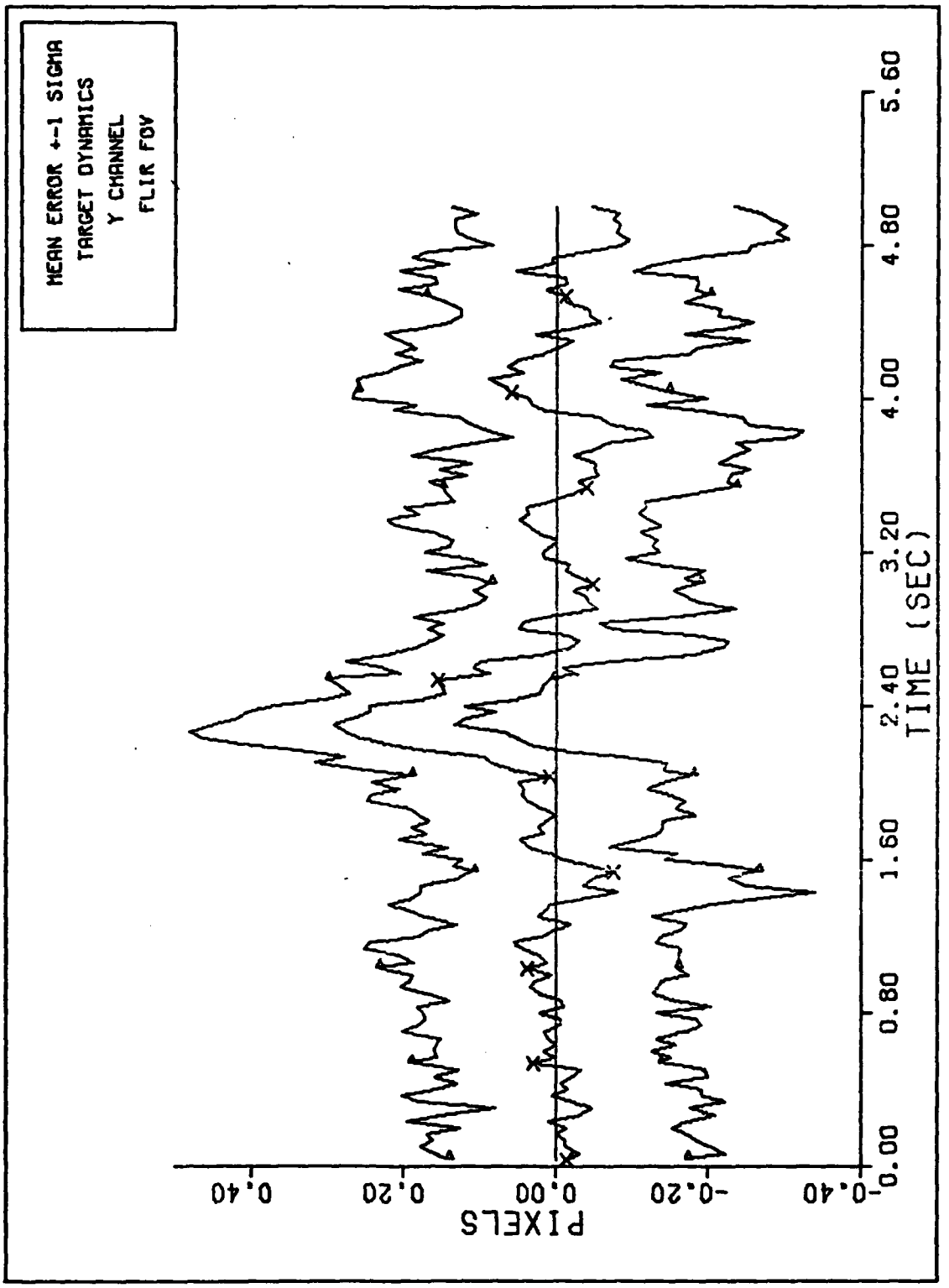
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure D-27 10 g Q=300 Performance Plot



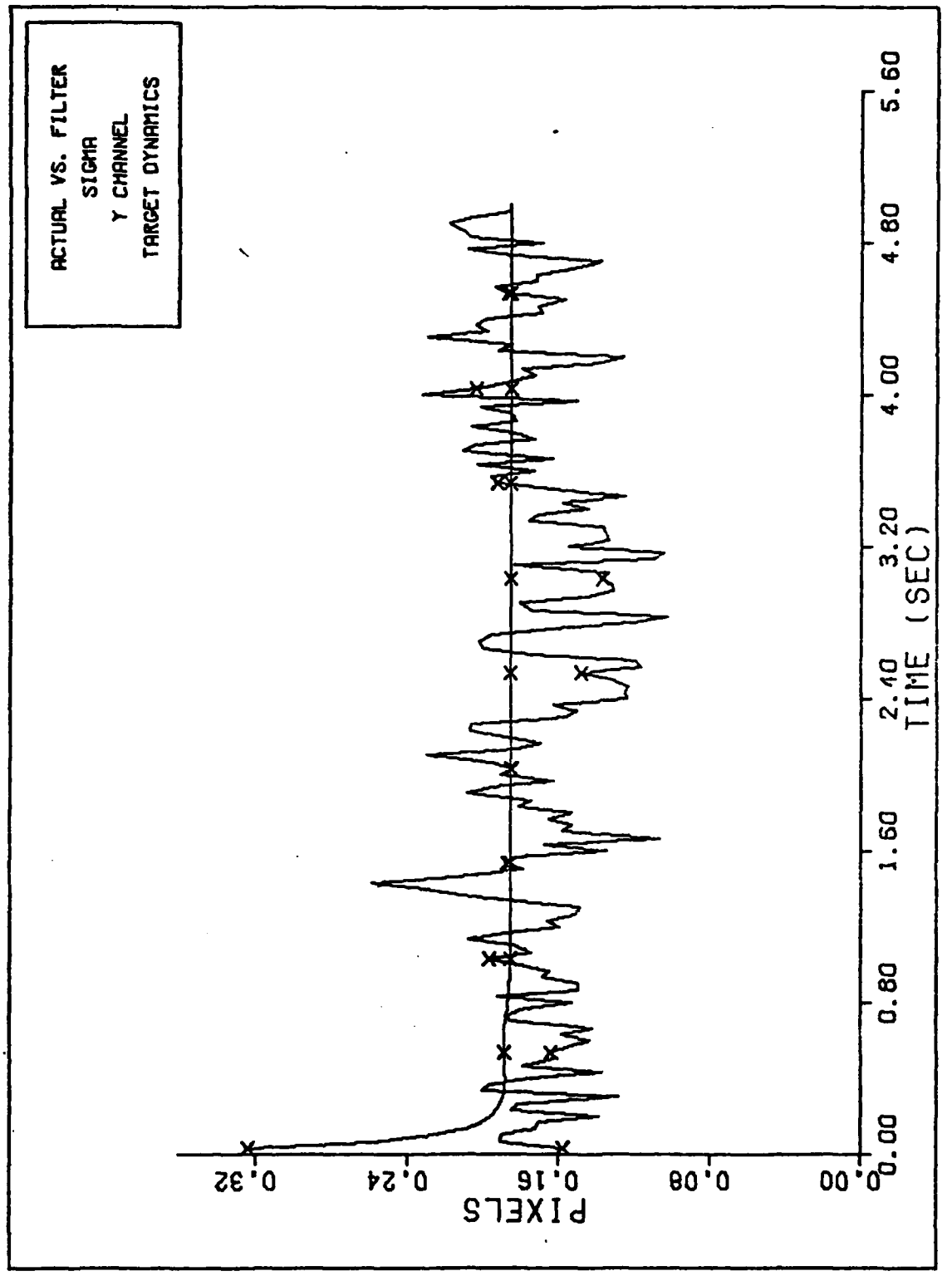
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-28 10 g Q=300 Performance Plot



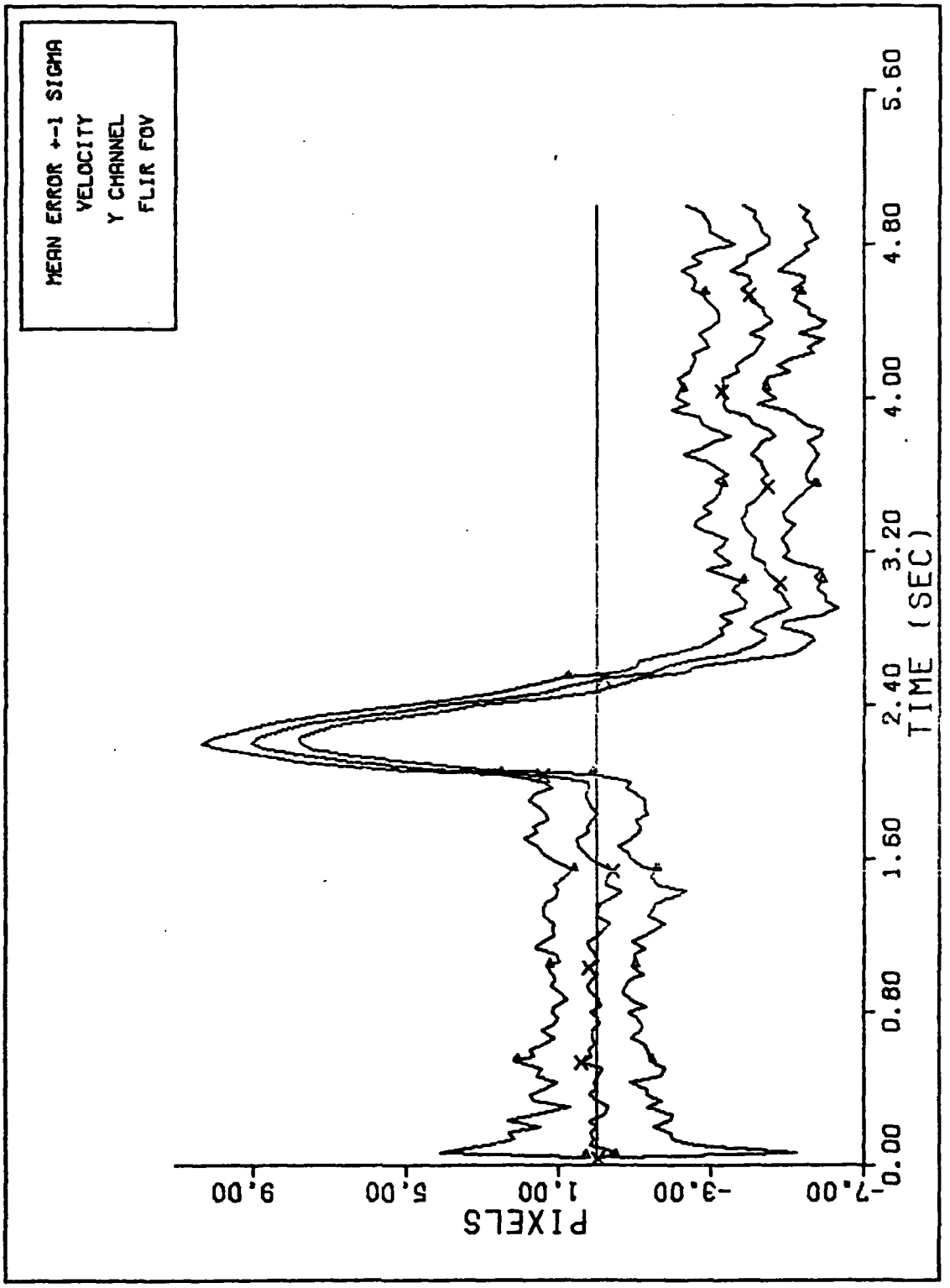
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure D-29 10g Q=300 Performance Plot



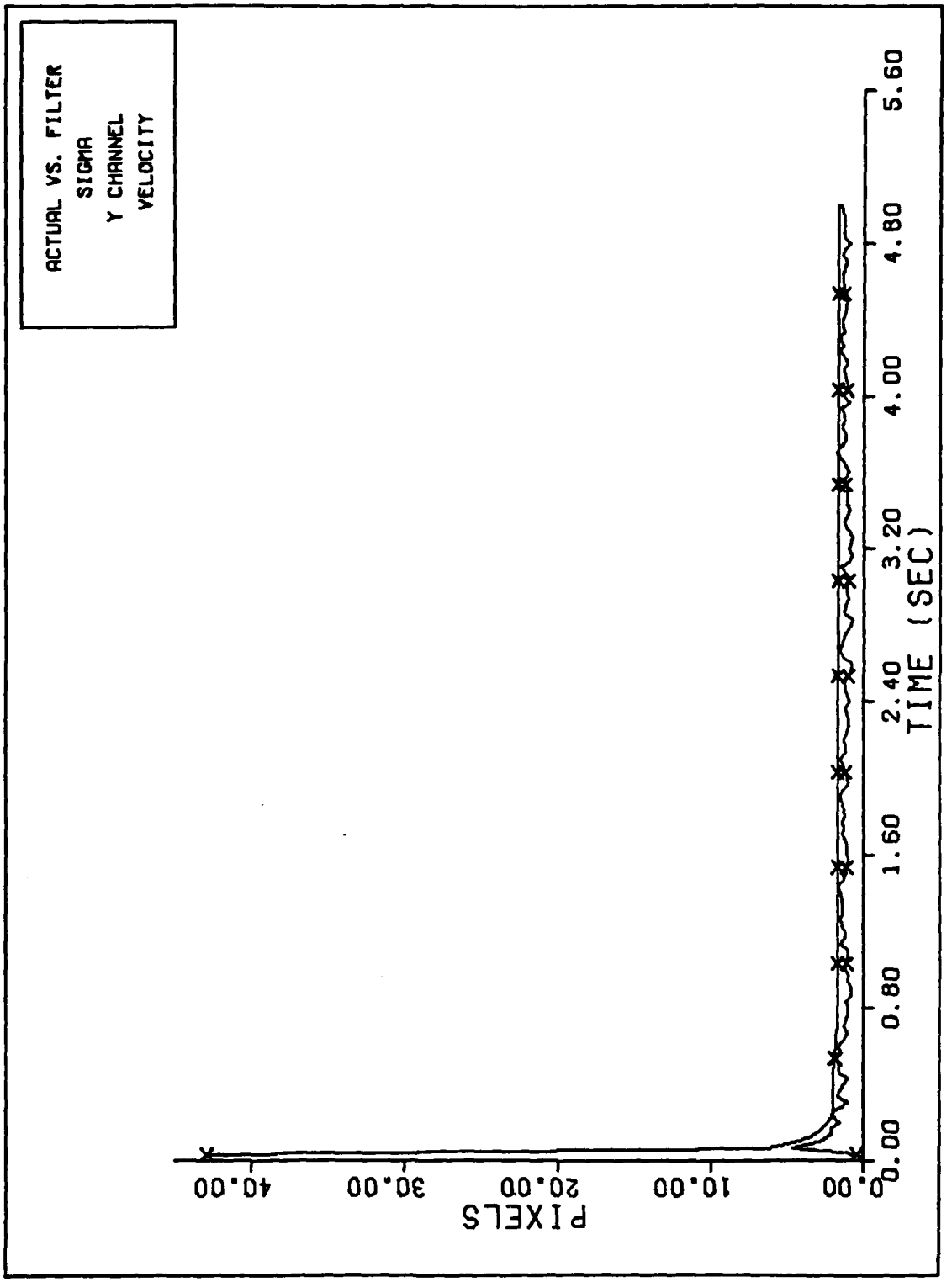
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-30 10 g Q=300 Performance Plot



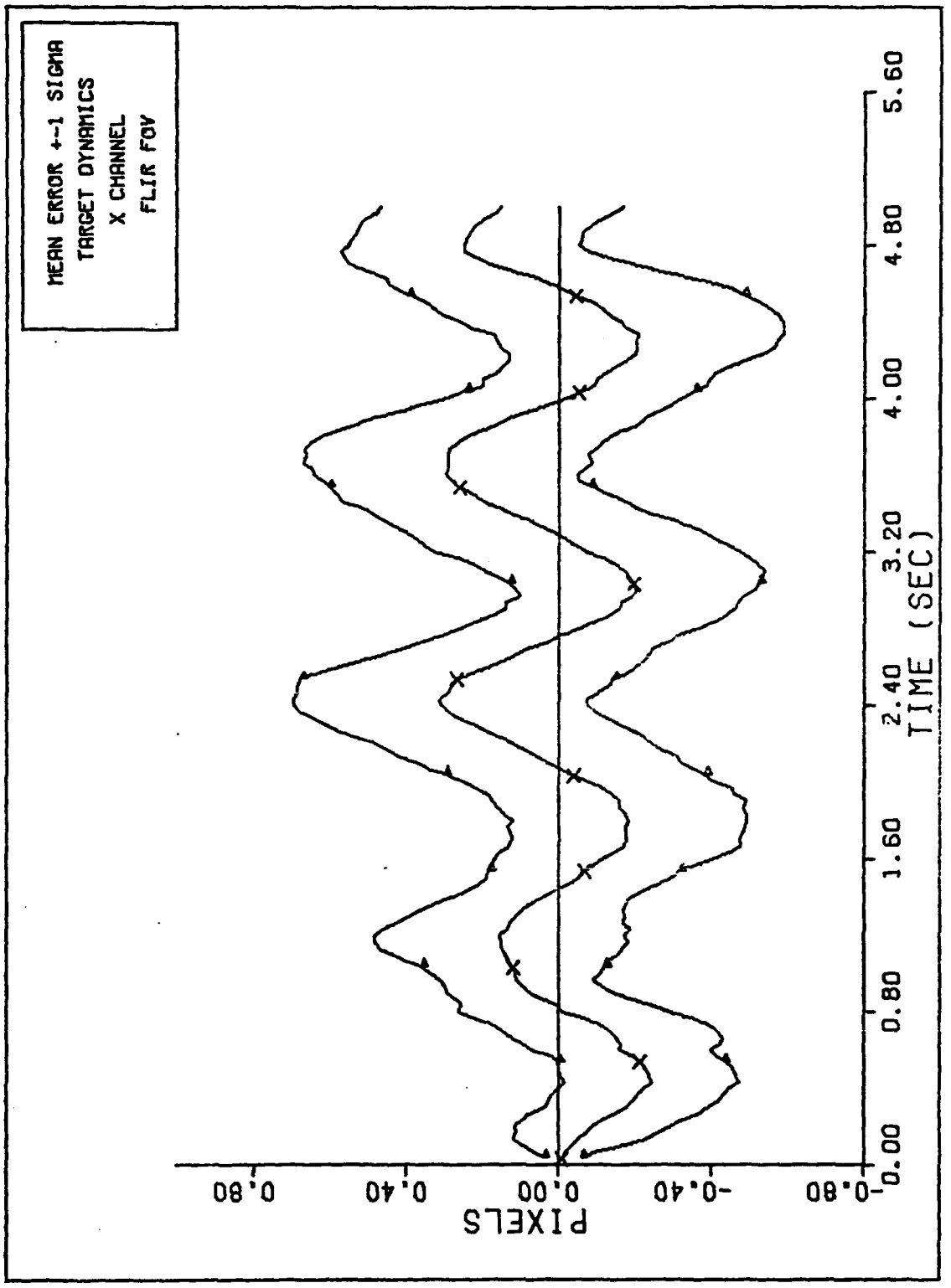
Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure D-31 10 g Q=300 Performance Plot



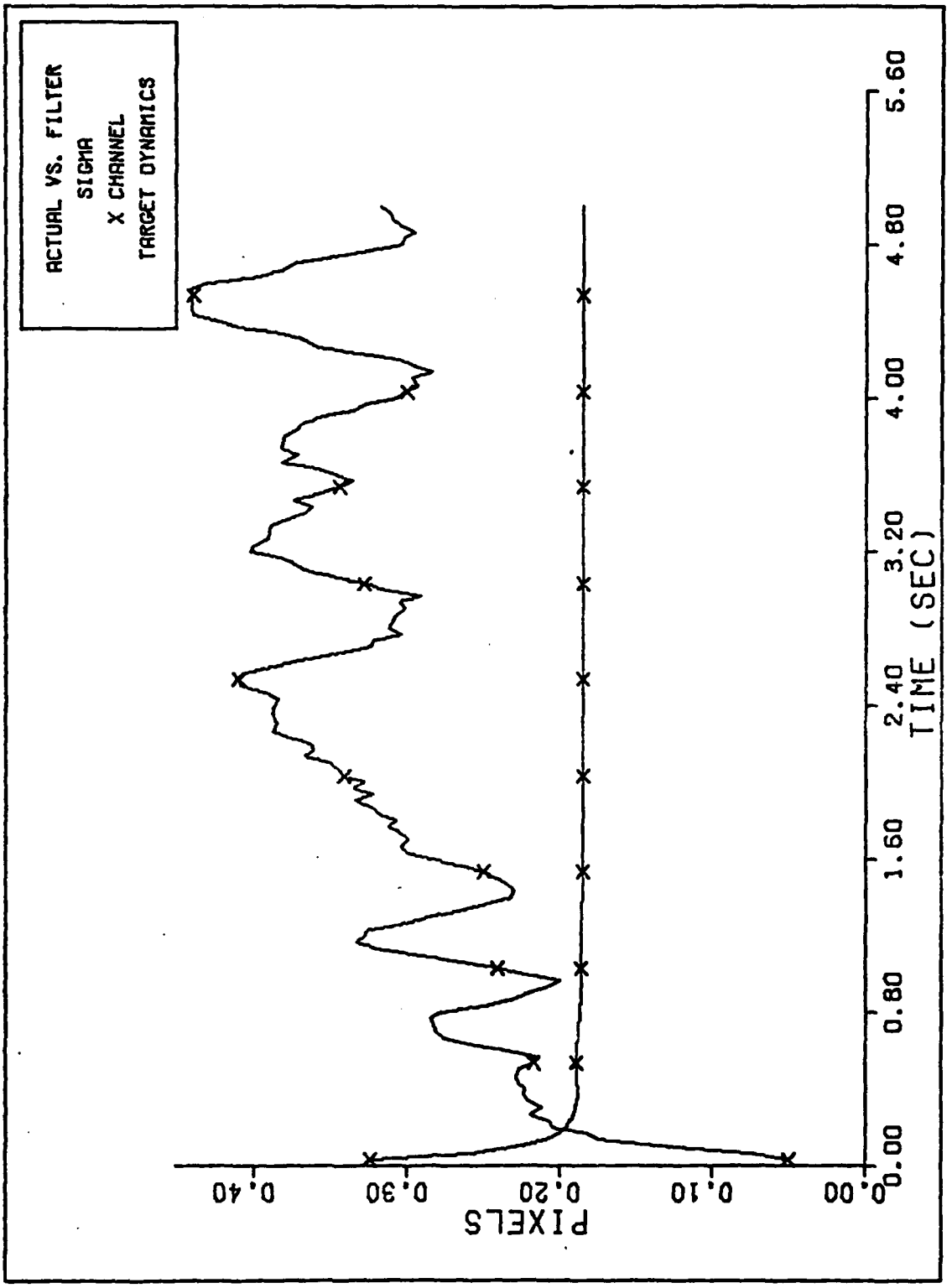
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-32 10 g Q=300 Performance Plot



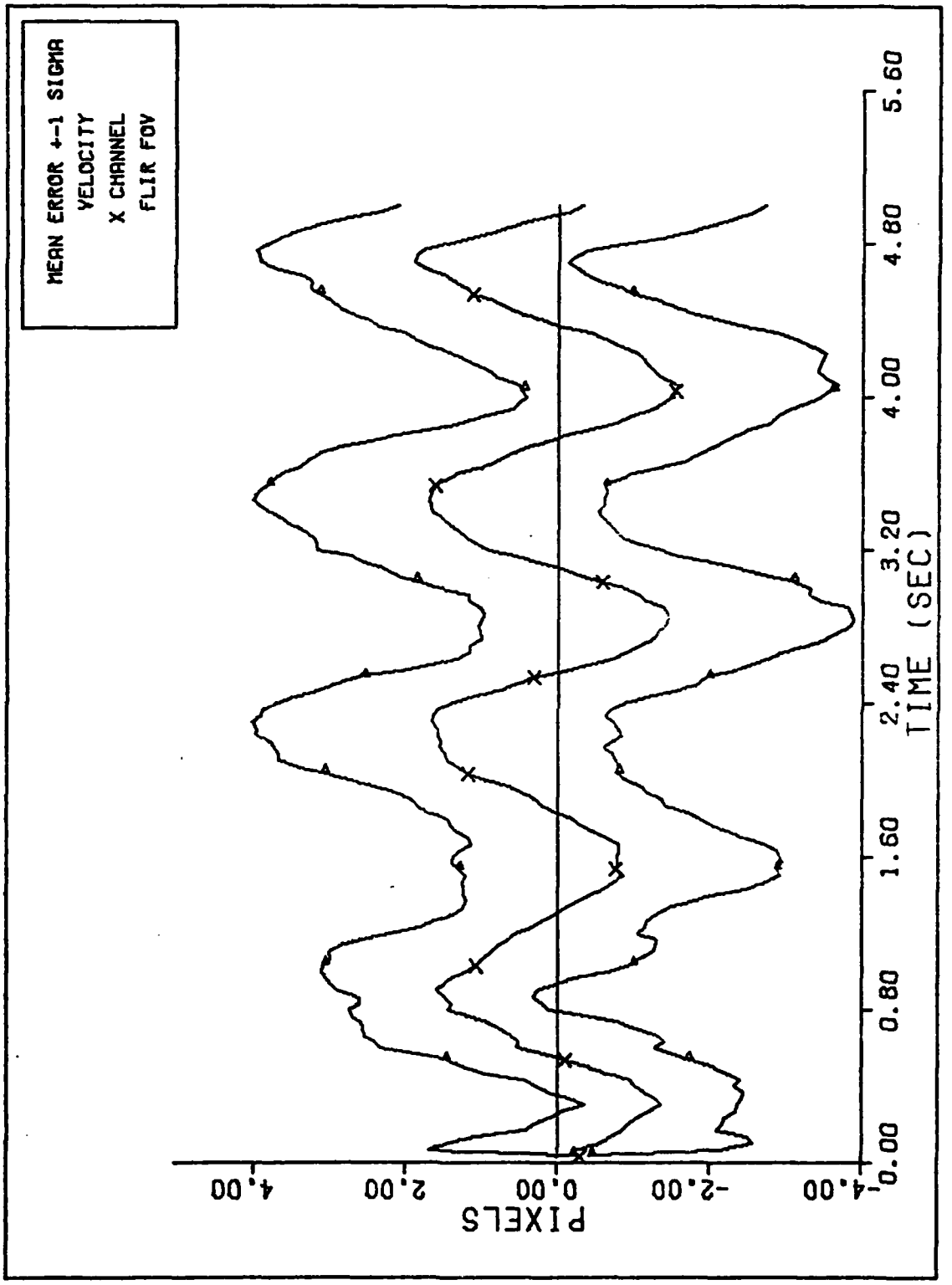
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure D-33 2 g Q=300 Performance Plot



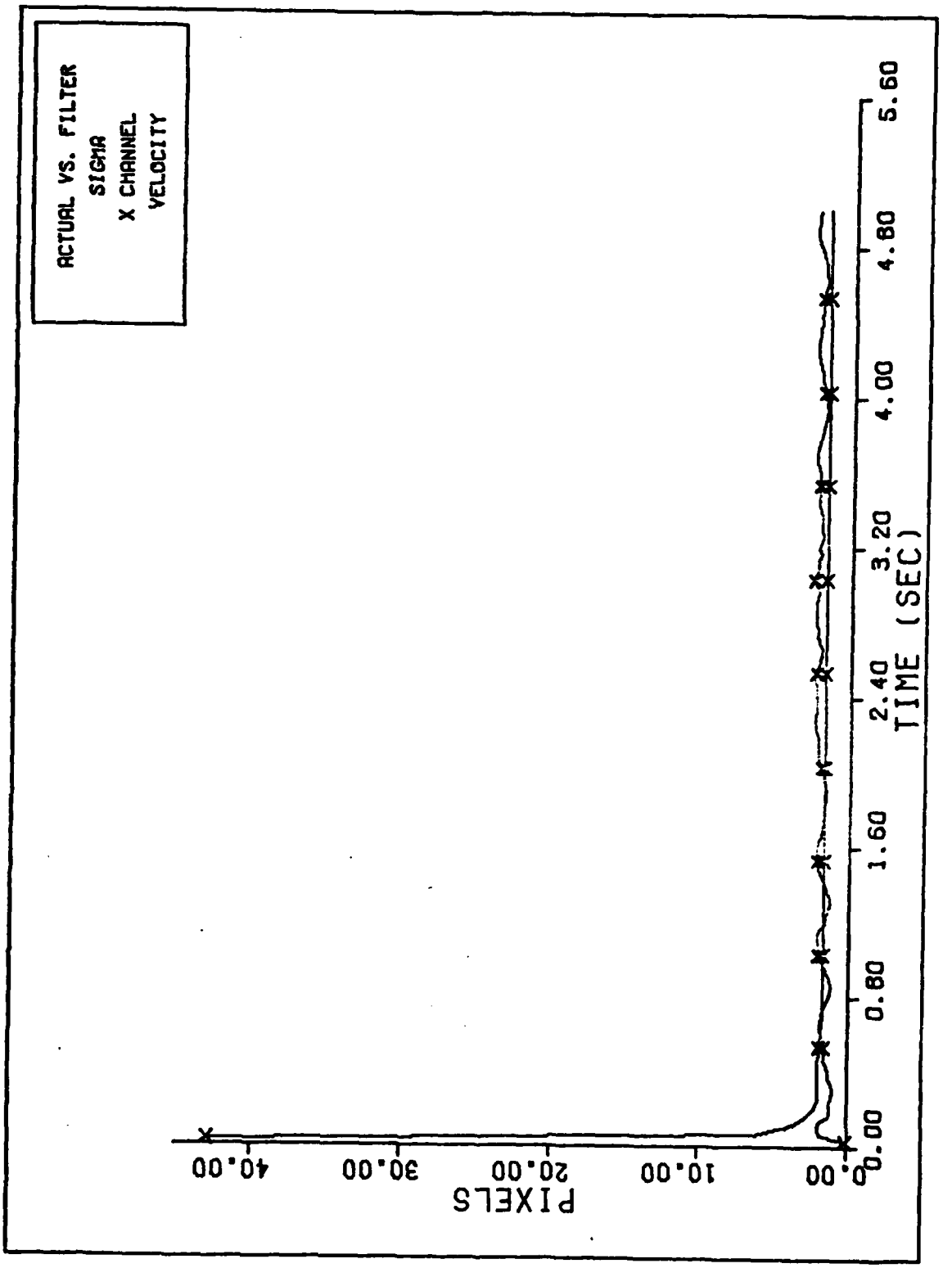
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-34 2 g Q=300 Performance Plot



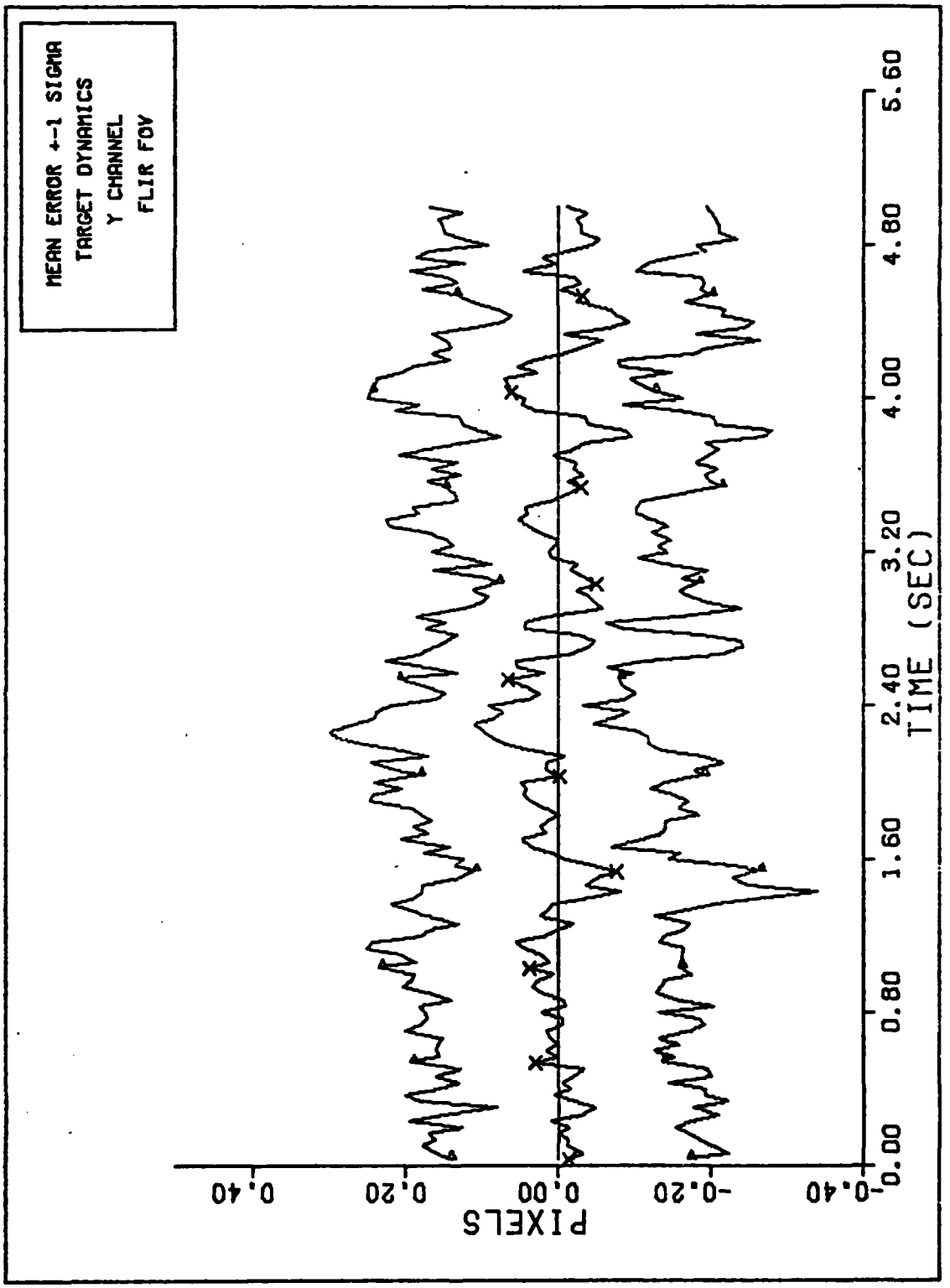
X CHANNEL VELOCITY ERROR (S/N= 12.5)

Figure D-35 2 g Q=300 Performance Plot



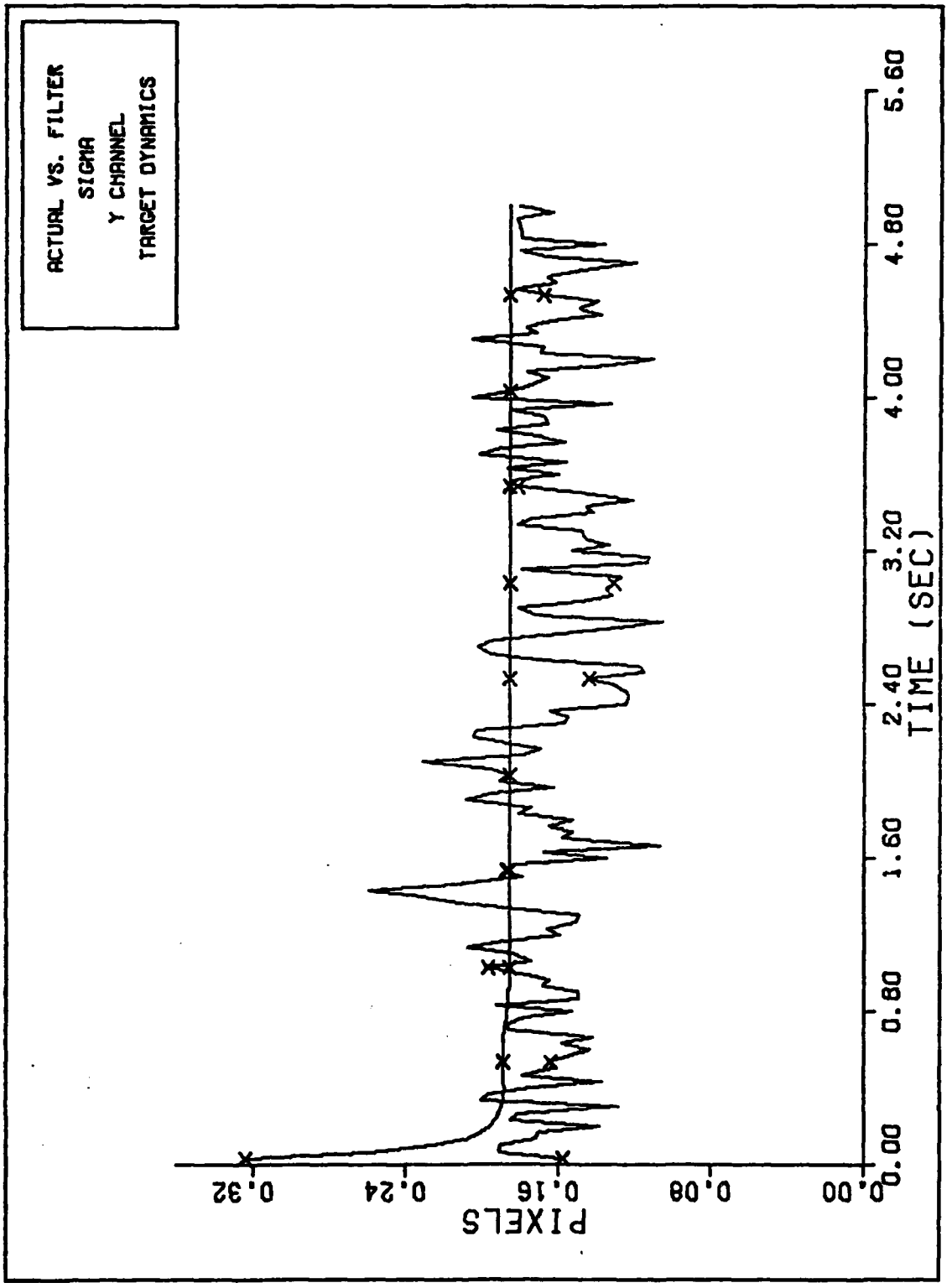
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure D-36 2 g Q=300 Performance Plot



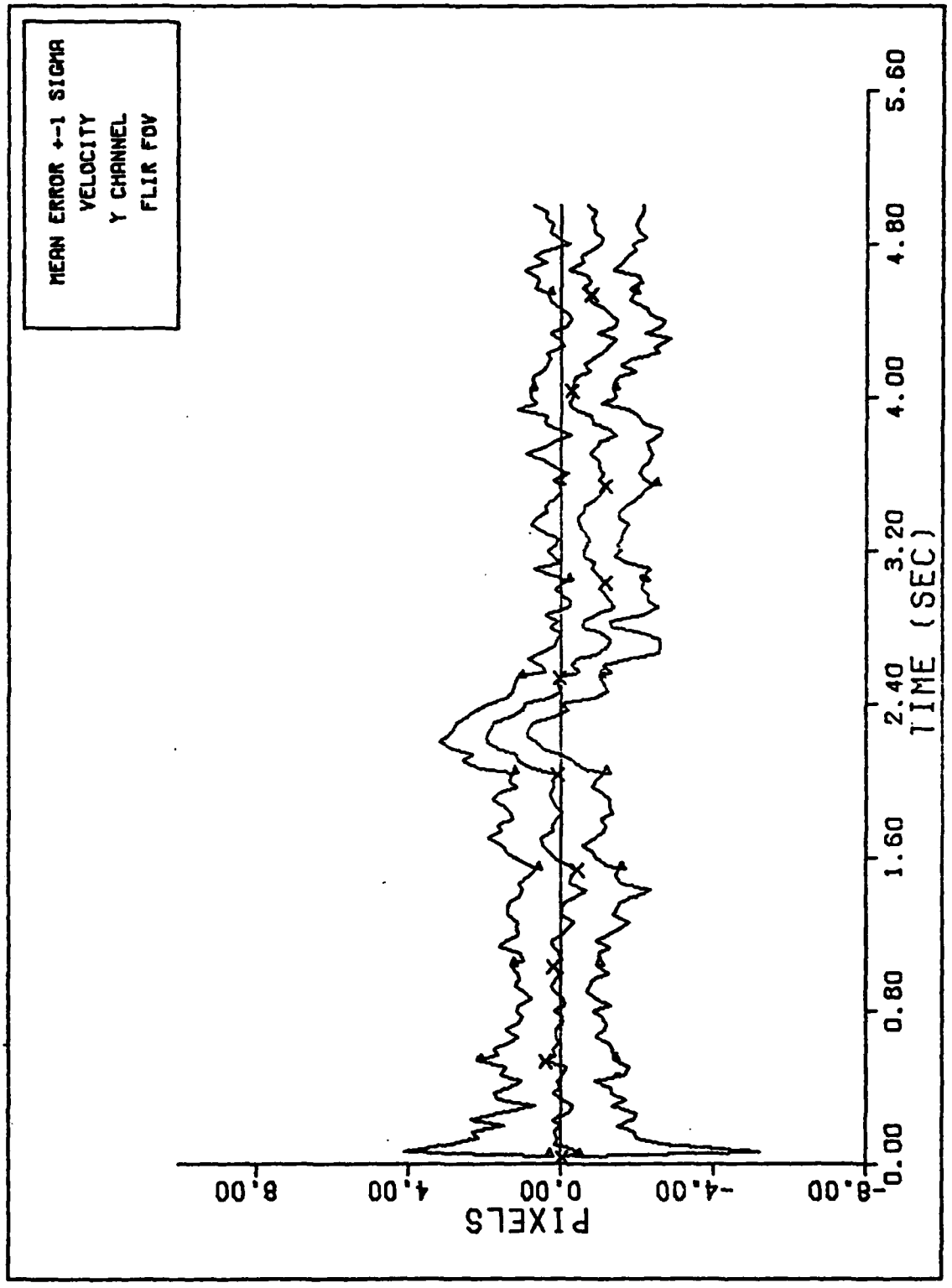
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure D-37 2 g Q=300 Performance Plot



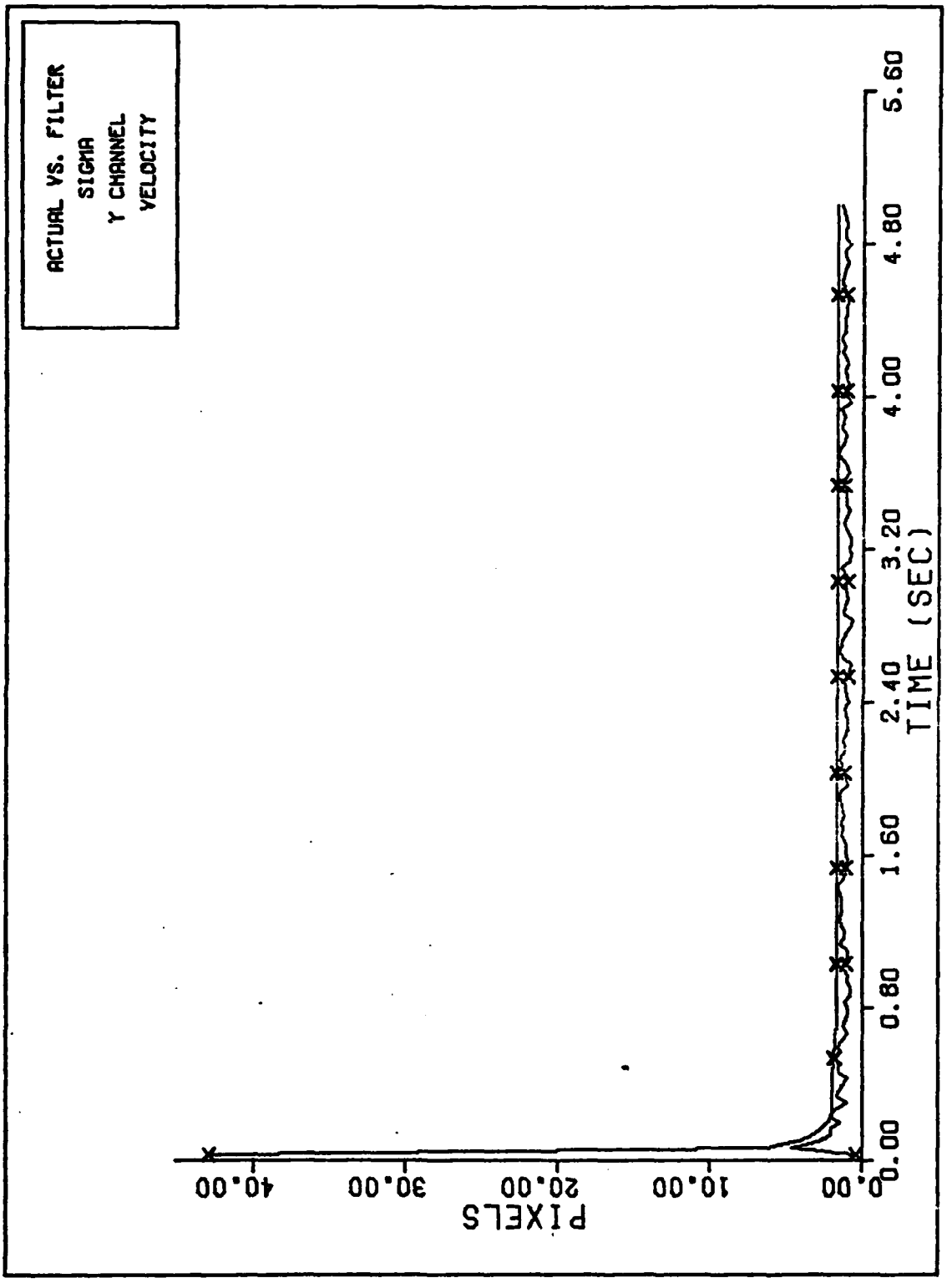
FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure D-38 2 g Q=300 Performance Plot



Y CHANNEL VELOCITY ERROR (S/N= 12.5)

Figure D-39 2 g Q=300 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure D-40 2 g Q=300 Performance Plot

Vita

Patrick M. Flynn was born on March 4, 1955 in Tacoma, Washington. He graduated from Clover Park High School in Tacoma, Washington in June 1973 and after two years at Tacoma Community College he enlisted in the United States Air Force. He was assigned to the AN/FPS-85 phased array radar site located at Eglin AFB, Florida. In August 1977 he received a ROTC scholarship. Lieutenant Flynn graduated with honors from Oregon State University, where he was a member of the Eta Kappa Nu electrical engineering honor society, in June 1980. He was then assigned to the Air Force Institute of Technology to pursue a Masters Degree in Electrical Engineering.

Permanent address: 92 West Shore Dr.
Tacoma, Wa 98498

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/81D-21	2. GOVT ACCESSION NO. AD-A115 509	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ALTERNATIVE DYNAMICS MODELS AND MULTIPLE MODEL FILTERING FOR A SHORT RANGE TRACKER		5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Patrick M. Flynn, 2Lt, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1981 13. NUMBER OF PAGES 160
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

15 APR 1982

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

18. SUPPLEMENTARY NOTES

Lynn E. Wolaver

Approved for public release; IAW AFR 190-17

~~Fredric C. Lynch, Major, USAF~~
~~Director of Public Affairs~~

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Extended Kalman filter
Target tracking
Multiple model
Constant turn rate

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The performance of three extended Kalman filter implementations that estimate target position, velocity, and acceleration states for a laser weapon system are compared using various target acceleration trajectories. Measurements available to the extended Kalman filters each update are taken directly from the outputs of a forward looking infrared (FLIR) sensor. Two dynamics models considered for incorporation into the filter are 1) Brownian motion (BM) acceleration model and 2) a constant turn rate (CTR

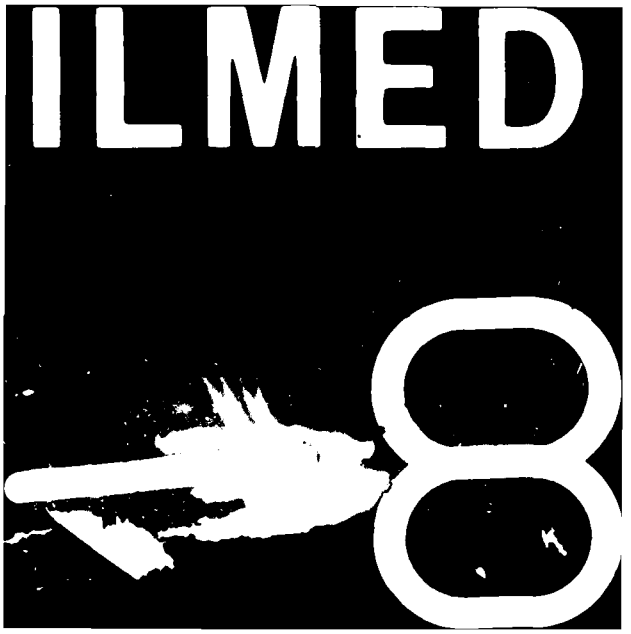
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

target dynamics model. The CTP filter was compared against the PM filter to see if the more complex dynamics of the CTP filter gave it a significant improvement in tracking performance over the PM filter. These two simple extended Kalman filters were then compared to a multiple model adaptive filter consisting of a bank of three filters based on the Brownian motion acceleration model. All three filters are tested using three different flight trajectory simulations: a 2 g, a 10 g and a 20 g pull-up maneuver. All evaluations are accomplished using Monte Carlo simulation techniques.

The constant turn rate extended Kalman filter was found to outperform the other two filters. The main advantage this filter had was the minimization of mean bias error in estimating position. The standard deviation of error was also slightly lower in most instances.


ILMED
8